NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

INTELLIGENCE DECISION SUPPORT SYSTEM
FOR THE REPUBLIC OF KOREA ARMY
ENGINEER OPERATION

by

Jang, Chang Ki

June 1987

Thesis Advisor Vincent Y. Lum

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION: Unclassified	1b RESTRICTIVE MARKINGS:
2a SECURITY CLASSIFICATION AUTHORITY:	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; Distribution is unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE:	
4 PERFORMING ORGANIZATION REPORT NUMBER(S):	5 MONITORING ORGANIZATION REPORT NUMBER(S):
6a NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable): Code 52	7a NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION: ; 8b OFFICE SYMBOL (If applicable):	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER:
8c ADDRESS (City, State, and ZIP Code):	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
INTELLIGENCE DECISION SUPPORT SYSTEM FOR THE REPUBLIC OF KOREA ARMY ENGINEER OPERATION (u)

12 PERSONAL AUTHOR(S)
Jang, Chang Ki

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year Month Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1987 June	87

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Intelligence System; Decision Support System; River Crossing Operation; Obstacle/Denial Operation

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

In the current circumstances of the Korean Army, the engineer units perform two major missions - river crossing and obstacle/denial operations - to support the combined arms team. To accomplish these missions, engineer units need various kinds of data and information during the planning phase of operations. This kind of data and information can be provided by the information processing system that is to be developed in this thesis.

This thesis is to apply a computer based information processing system to the planning phase of the military operations. The author presents an intelligent decision support system for the Army Engineer Operations, specifically the river crossing and obstacle/denial operations. The purpose of this system is to maintain and analyze the

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified	
22a NAME OF RESPONSIBLE INDIVIDUAL: Dr. Vincent Y. Lum	22b TELEPHONE (Include Area Code): (408) 646-2449	22c OFFICE SYMBOL: 52Lu

#19 ABSTRACT (Continued)

related information stored in the computer and to provide the resulting information to the commanders and their staffs to help them make their decisions more effectively. To accomplish this task, the author has used the structured system analysis and design methodology through the system development process, and has implemented the river crossing operation in a microcomputer with dBASE III Plus as an example.

Approved for public release; distribution is unlimited.

INTELLIGENCE DECISION SUPPORT SYSTEM FOR THE REPUBLIC OF KOREA ARMY ENGINEER OPERATION

by

Jang, Chang Ki
Major, Republic of Korea Army
B.S., Korea Military Academy, 1976

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1987

ABSTRACT

In the current circumstances of the Korean Army, the engineer units perform two major missions - river crossing and obstacle/denial operations - to support the combined arms team. To accomplish these missions, engineer units need various kinds of data and information during the planning phase of operations. This kind of data and information can be provided by the information processing system that is to be developed in this thesis.

This thesis is to apply a computer based information processing system to the planning phase of the military operations. The author presents an intelligent decision support system for the Army Engineer Operations, specifically the river crossing and obstacle/denial operations. The purpose of this system is to maintain and analyze the related information stored in the computer and to provide the resulting information to the commanders and their staffs to help them make their decisions more effectively. To accomplish this task, the author has used the structured system analysis and design methodology through the system development process, and has implemented the river crossing operation in a microcomputer with dBASE III Plus as an example.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

In modern society, a great deal of information is disseminated in our daily life through the news media such as newspaper, magazine, and television or radio broadcasting. We need information for various purposes. However, it is hard to get the desired information, particularly at the time when we need it. Moreover, it is impossible to remember all of the information that we have ever received. Fortunately, the advance of the computer technology has facilitated much this particular problem. The computer can store and retrieve vast amounts of information, that can be used to an advantage.

Information processing is a major activity in today's society. A significant part of an individual's working and personal time is spent recording, searching for, and absorbing information. Thus, computers have become an essential part of the individual or organizational information processing. The current challenge is how to gain the maximum benefits with the use of the computers in different activities, including managerial tasks and decision making.

The ancient Chinese strategist ***Sonja*** said in war, *"We can win 100 times in 100 battles if we have all the information about our enemy and ourselves."* This exemplifies the importance of information for military operations. For example, during World War II, the combined forces made the turning point in the European front with the Normandy Landing Operations. At that time, Hitler and his staff had not enough information about the combined allied forces. They believed that the allied forces would land at the coast of Pas De Calais in France because Pas De Calais is the nearest coastal area from England and they thought that the combined allied forces have not enough troops, fleets and aircrafts. But the combined forces had enough troops, fleets and aircrafts for the landing operations. They also had enough information about the terrain of the alternative landing area and the enemy's course of action. They succeeded because they conducted the landing operation at the Normandy coast while Hitler prepared strong defense along the coast of Pas De Calais, and a weak defense along the coast of Normandy.

In the modern military situations, we need various kinds of information when we prepare an operation. Information about the capability of the enemy and the friendly

forces, about the terrain and weather of the operation area, and about the enemy's course of action is extremely valuable. Most of the times, commanders have to do analysis and evaluation of the information quickly. This is necessary because in the modern battle fields, both the friendly and the enemy forces are highly modernized and their combat powers are devastating and their movements are rapid.

The basic idea of this thesis is to apply computer based information processing system to the army engineer operation in a military operation. Two major engineer operations, river crossing operation and obstacle/denial operation, are analyzed in detail. During the engineer operations, a commander and his staff have to analyze a large amount of information related to the operations within a very limited amount of time. In the proposed system of this thesis, information and data will be stored in the computer for consistency, accuracy and ease of access. Furthermore, to help the users make decisions fast, some of the processes in which decisions are derived will be done by the computer with the results fed back to its users. In other words, the goal of my thesis is to develop a microcomputer-based intelligence decision support system to increase the operational capability of the commanders and staffs.

A decision support system(DSS) is a coherent system of computer based technology used by managers as an aid to their decision making in semi-structured decision tasks. Decision support systems allow the decision maker to retrieve data and test alternative solutions during the process of problem solving. The decision support system should provide ease of access to the database containing relevant data and interactive testing of solution. Expert systems are a type of decision support system that incorporate the knowledge base and heuristics of an expert with a flexible interface so that even a novice can use the system to solve problems. The primary requirements of decision support for intelligence is the ability to search the database for opportunities and problems. In order to design a DSS, the designer must understand the process of decision making for each situation.

Chapter II discusses the background which relates to the military operation, especially the army engineer operation concepts in various situations. In Chapter III, we first discuss the system analysis and requirements. Problem definition, description about existing system, and requirements specification are done in this chapter. In addition we will develop the high level data flow diagram and functional decomposition of the logical model. In Chapter IV, we discuss the system design process and the implementation of the proposed system. Finally, in Chapter V, conclusions and recommendations are presented.

II. BACKGROUND

A. ARMY OPERATION CONCEPTS

1. Introduction

The goal of all operations is to destroy or hold off the opposing force. At the foundation of the Army's operations are the principles of war and their application to classical and modern warfare. An army unit will conduct all types of operations to preserve its status and to exploit the enemy's weaknesses. It will attempt to achieve its goal through the use of fire power or by out-maneuvering its enemy. It will maintain the agility necessary to shift forces and fires to be directed to the enemy's weak sides. To achieve its goal, an army's operations must be rapid, unpredictable, devastating, and disorienting to the enemy. The pace must be fast enough to prevent him from taking effective counter actions. Operational planning must be precise so that the combined arms operation in a battle is well-coordinated. It must also be sufficiently flexible to respond to changes or to capitalize on fleeting opportunities to exert maximum damage to the enemy.

2. Offensive Operations

The military operation is divided into two large groups: offensive operations and defensive operations. Offensive operations are characterized by aggressive initiatives on the part of the subordinate commanders, by rapid shifts in emphases to take advantage of opportunities, and by the use of the destructive power to bring destruction to the enemy's defenses as rapidly and as extensively as possible.

The offensive operations are undertaken to achieve several purposes. As destroying the enemy's fighting force is the only sure way of winning, offensive operations are primarily intended to destroy the enemy's forces. However, it is not necessary to defeat every enemy combat formation to win. Attacks that avoid the enemy's main strength but shatter the will of the opposing army or to reduce its fighting capability are the fastest and the cheapest way of winning. Offensive operations also have secondary purposes, all of which contribute to destroying the enemy. Elements of large attacking forces may undertake offensive operations specifically to secure key or decisive terrain because it has a serious effect to a combat situation, and as we know, the situation of resources has in the past determined the

outcomes of many wars. Therefore to deprive the enemy of resources is one of the purpose of offensive operations. In addition we can gain information about the enemy's current situation through offensive operations. And we can deceive and divert the enemy's attentions to irrelevant or unimportant matters. Also we can hold the enemy in position during the preparation of an offensive operation to allow us to attack the enemy from other directions. All of these, plus others, serve as objectives of offensive operations. [Ref. 1: p. 8-4]

3. **Defensive Operations**

Whether an offensive operation is a success or a failure, we have to prepare defensive operations afterward because they are important to retain the gain of the offensive operations and to prepare the next phase of an offensive operation. Some military theorists think defense is the stronger form of war because denying the enemy's success goes before achieving our own success. Indeed, the defenders have significant advantages over the attackers. In most cases they not only know the ground better, but having occupied it first, have strengthened their position. Therefore we conduct defensive operations to achieve several purposes. Throughout a defensive operation, a defender can block an enemy's attack and concentrate forces elsewhere to attack enemy's weak points while holding up the enemy forces in front of his defense position. Further he can control the key terrain to the advantage of the defensive or offensive operations on his side . Besides, it also take more offensive power to overcome a defense; thus his defensive operation generally can tie down more of the enemy's manpower. [Ref. 1: p. 10-3]

4. **Sequence of Commander and Staff Actions**

The sequence of actions in making and executing a decision involves a series of separate actions or steps. As shown in Figure 2.1, a higher headquarters normally assigns the mission to the subordinate units, but the commander of a unit usually develops or expands the mission. The commander may initiate his mission analysis at this point. After that his staff provides the commander the available relevant information. Based on this information, the commander completes his mission analysis and issues his planning guidance to his staff. The purpose of the mission analysis is to insure that the commander fully understands his mission and to allow him to develop those tasks that are essential to the accomplishment of the mission. At this time, the commander normally includes in his initial guidance the mission restated appropriately as determined by his mission analysis [Ref. 2-5: p. 5-13].

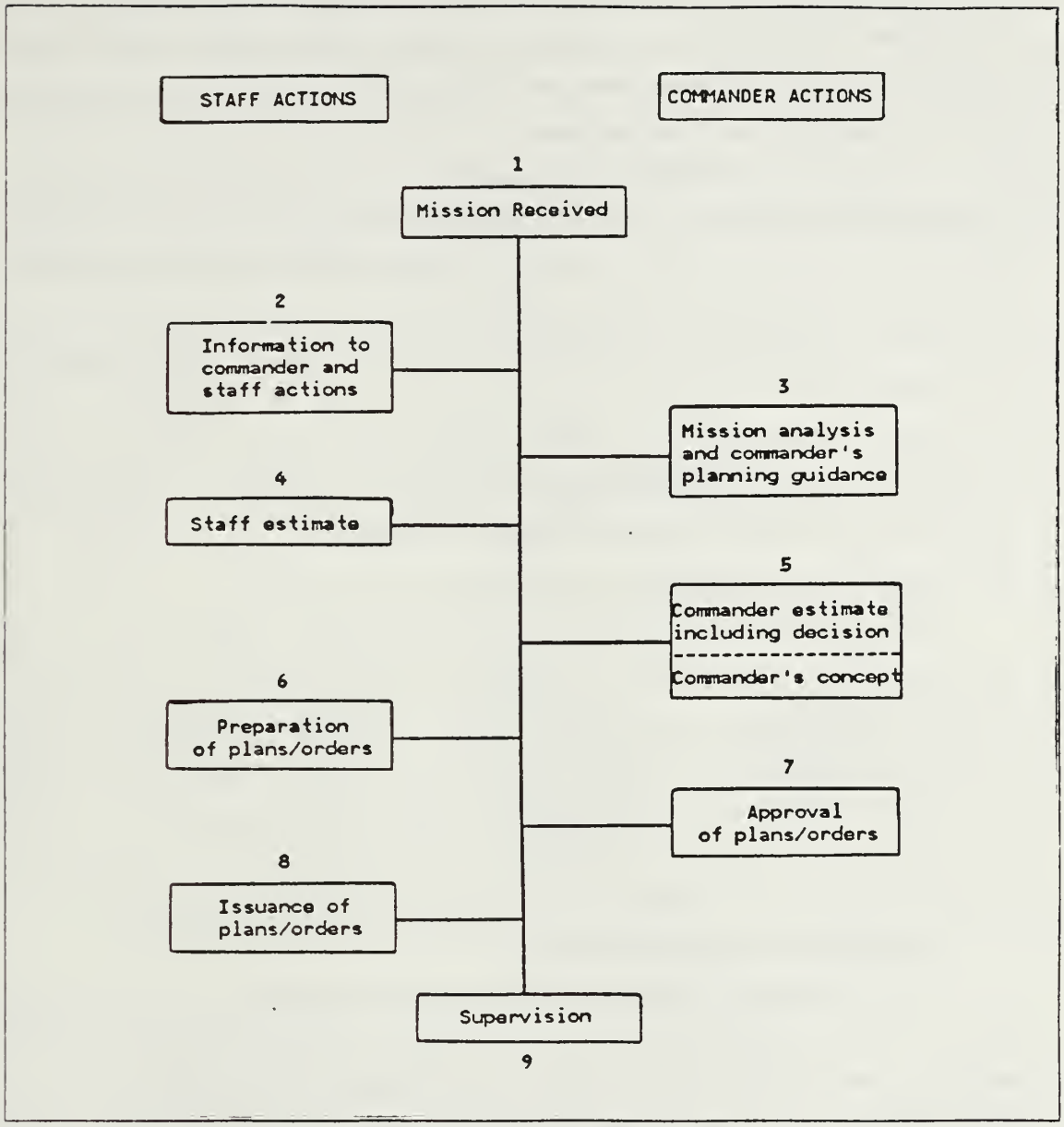

Figure 2.1 Sequence of Commander and Staff Actions.

Based on the restated mission and planning guidance received, the coordinating staff officers who may also prepare their own estimates when required, prepare their staff estimates with the assistance of the special officers. In turn, the coordinating staff officers present their final estimates to the commander, recommending the actions that the commander should take to accomplish the mission. After presenting the recommendation, the commander considers the recommendations

of his staff, completes his own estimate, and announces his final decision. Following the decision statement, the commander may provide the staff his overall concept of how the operation will be conducted, which is generally an amplification of his decision along with some necessary explanation.

Based on a complete understanding of the commander's decision and the concept of the operation, the staff members determine the actions, including the preparation of plans or orders required by the command to carry the operation to a successful completion. Normally the staff submits plans and orders to the commander for approval before they are published as plans or orders. After publishing and distributing plans or orders to the surbodinate units, the command's and staff's supervision of the execution of orders is a continuous action. Therefore, whenever a commander receives the mission from his higher headquarters, a plan or order is generated by a sequence of commander and staff actions.

B. ENGINEER SYSTEMS IN THE ARMY

Today more than ever before, the engineer units play a critical role as a member of the combined arms team and they fight as an integral part of it. As movement and intensity on the battle field increase, the requirement to reinforce a position complementing the natural terrain increases. The engineer brings to the combined arms team a terrain-oriented system that enhances the capability of our weapon systems while decreasing the effectiveness of the enemy's weapons. The idea is to employ the engineer unit as far forward as possible with the fighting units.

1. Organization of Engineer Systems

In the Republic of Korea Army(ROKA), the engineer systems are organized under the Division, Corps and Field Army Commands. But each level of command has a different type of engineer units and their missions, capabilities, and equipments are a little bit different from each other. An engineer unit divides into division engineer, corps engineer, and field army engineer.

a. Division Engineer

Each infantry and mechanized infantry division has an original engineer battalion. A division engineer batalion consists of a Headquarters and a Headquarter Company(HHC), three combat engineer companies, and a bridge company as shown in Figure 2.2. The HHC is organized into the normal staff sections and an equipment platoon. Generally each combat engineer company has 5 officers and 144 enlisted men organized into a Company Headquarters(HQ) and three platoons. The bridge

company has 5 officers and 146 enlisted men organized into two heavy raft sections, one armored launch bridge(AVLB) section and a company HQ. The heavy raft sections have a M4T6 float bridge which is a river crossing equipment to move across the heavy combat equipments such as tanks and armored vehicles by bridging or rafting. The mission of the division engineer battalion is to increase the combat effectiveness of the division, and to carry out an infantry mission when required by the division commander [Ref. 3: p. B-8].

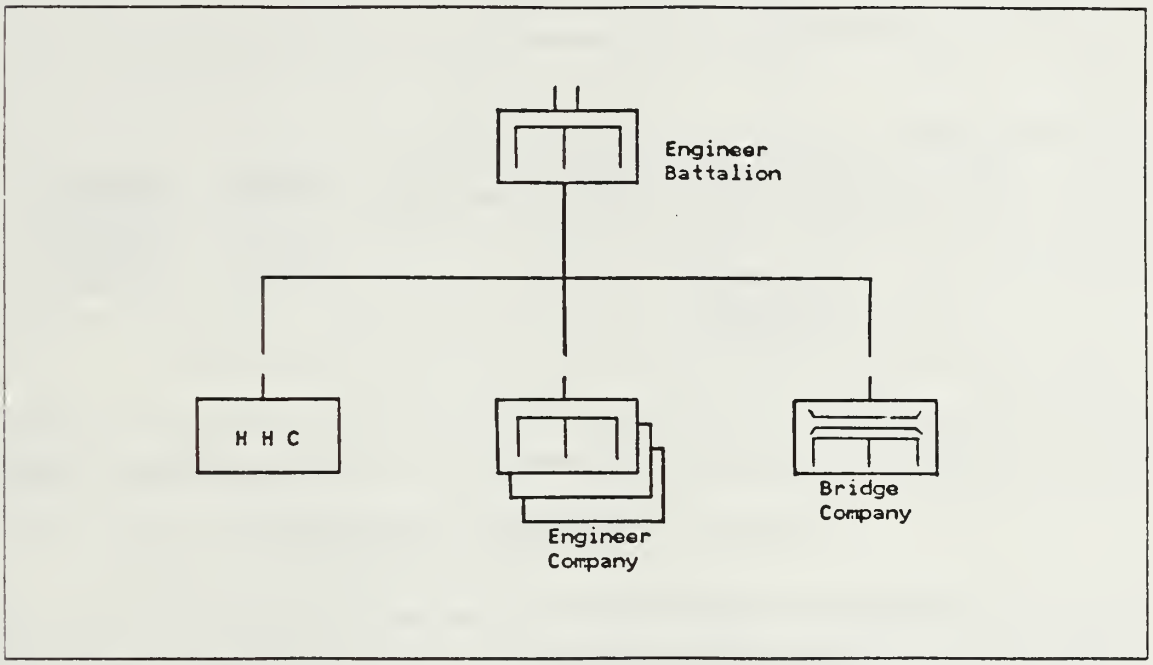

Figure 2.2 Organization of Division Engineer.

The mission capability of a division engineer battalion is to emplace and remove obstacles like mines and boobytrap, to make possible either carefully or hastily planned river crossings with boats, rafts, and temporary bridges, and to construct, repair, and maintain roads, bridges, landing strips and helipads. Further, the duties of a division engineer battalion is to assist in the assault of fortified positions, to install and operate portable water supply facilities, and to provide reconnaissance to gather engineer intelligence.

b. Corps Engineer

Engineer brigades are organized under each corps command that consists of three or more divisions. An engineer brigade consists of three to five engineer

battalions and a heavy equipment company, a panel bridge company, a dump truck company, and a floating bridge company as shown in Figure 2.3.

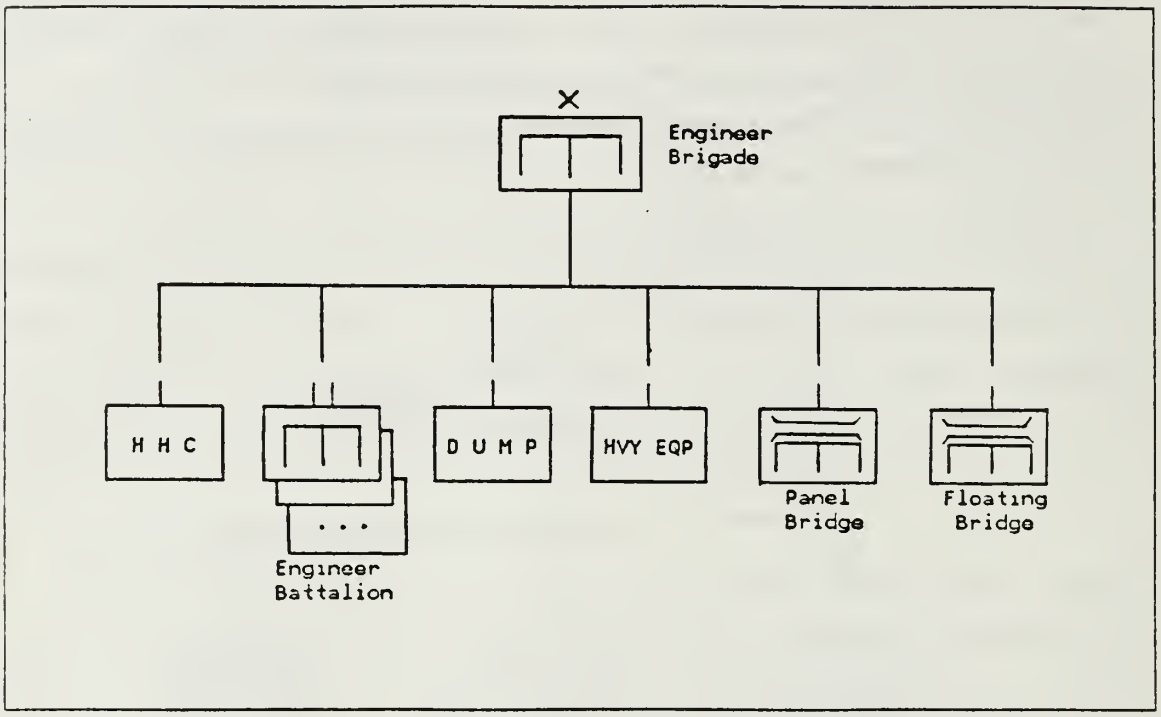

Figure 2.3 Organization of Corps Engineer.

A battalion consists of HHC and four line companies. The HHC consists of the normal staff and company sections plus an equipment platoon, and a combat construction section(plumbers, carpenters, and other skilled construction trades). Each line company consists of 5 officers and 139 enlisted men organized into a company HQ and three platoons. Each battalion has the same organization as a division engineer battalion but the mission of corps engineer battalions that are organized under an engineer brigade, is to increase the combat effectiveness of the corps by means of engineering combat support and general engineering work, to reinforce the divisional engineer units when required, and to perform infantry combats as necessary. The responsibilities of corps engineer battalion, like those of the division engineer, is to construct, and repair. A commander of corps can operate any engineer battalion or engineer company organized under an engineer brigade or corps command [Ref. 3: p. B-15].

c. Field Army Engineer

Each Field Army Command in the ROKA has two to four corps commands, and they also have engineer units, called field engineer groups. An engineer group is similar to an engineer brigade but it's a little bit smaller.

An engineer group consists of three to four engineer battalions and a light equipment company, a panel bridge company, a dump truck company, and a floating bridge company. Each battalion is organized into an HHC and three engineer companies. Each engineer company consists of 6 officers and 186 enlisted men organized into a company HQ, a horizontal construction platoon, and two general construction platoons. [Ref. 3: p. B-16]

The mission of an engineer group is to construct and rehabilitate roads, airfields, pipeline system, structures, and utilities for the Army and Air force, to assist in emergency recovery operations, to increase the combat effectiveness of the division, corps, and army group forces by means of engineering combat support and general engineering work, and to perform infantry combat mission, when required.

The capability of an engineer group includes the construction or rehabilitation of routes of communications, bridges, forward tactical and forward cargo airfields and heliports, and to provide limited reconstruction of railroads, railroad bridges, and ports of the corps rear area.

2. **Objectives of Engineer Systems**

In any kind of army operations, an army must contain its the engineer system the skills and equipment to provide mobility and survivability to friendly forces, counter mobility to the enemy, and to accomplish general engineering work to friendly forces. [Ref. 3: p. 2-2 - 2-6]

a. Mobility

In modern battle fields, to be more effective than our enemy, we must move into and out of battle positions faster than our enemy. Mobility is achieved through reducing or negating the effects of existing obstacles to facilitate the movement of maneuver/fire units and critical supplies. Thus, the engineers' work include breaching the obstacles created by the enemy's counter actions such as minefields, road craters, and ditches, by using their organized equipments to make quick bypass around them when it's not possible to breach the obstacles, and bridging the gaps with their equipment or with timber that can be obtained around the battle field.

b. Counter Mobility

The enemy will try to improve the advance of fire power and maneuver units to destroy the friendly forces or to secure a key terrain for the next operation. Therefore, friendly forces have to reduce the enemy's advance effectively. In order to limit the enemy's movement, we do various kinds of counter actions like emplacing obstacles such as minefields, road craters, abatis, and road block etc., as dictated by the battle field with whatever available resources. Also engineer units demolish road structures such as bridges, tunnels etc. that can seriously hamper the enemy's advance.

c. Survivability

Survivability is the development of protective positions. Normally, positions are expedient in nature, providing only side and frontal protection against direct and indirect fire weapon systems. The greatest number of protective positions are constructed in the defensive because the fire power of modern battle fields is very destructive. The engineer units provide the specialized equipment and expertise to assist maneuver units to increase their survivability in battle field.

d. General Engineering

General engineering is the support of units and activities in the brigade and division rear areas. The division engineer battalion is staffed and equiped to work for the committed maneuver brigades. Corps engineers, who are supporting the division, provide the primary capability to accomplish general engineering tasks in the division area. General engineering missions are improving and maintaining essential combat and main supply routes, replacing assault or blown bridges with tactical bridging, clearing minefields, providing potable water, and providing terrain studies etc. [Ref. 3: p. 3-32]

3. **Engineer Operations**

As already mentioned before, the engineer system provides mobility, counter mobility, survivability, and general engineering as a member of the combined arms team during the Army operations [Ref. 3: p. 2-2]. To accomplish their own missions, engineer operations are divided into three large groups such as river crossing operations, obstacle creation and denial operations, and recovery operations.

a. River Crossing Operations

Rivers, lakes, and canals are critical natural obstacles hindering maneuvers during offensive and defensive operations. We plan and conduct river crossing operations in order to move combat power across these obstacles. The objective of

river crossing operations is to project combat power across a water obstacle while retaining the integrity and momentum of the force.

Engineer units conduct river crossing operations as a member of the combined arms team. In the planning phase of a river crossing operation, engineer units have to provide information to their commander about available crossing sites and their characteristics and available river crossing equipments to support the combined arms team. In the executing phase of the river crossing operation, the engineer units use river crossing equipments to build floating bridges or rafts, and operate these equipments.

A river crossing is a special operation in that it requires substantial amount of planning and support. Two types of crossing are conducted, the hasty and the deliberate crossing. Which type of crossing is to be selected depends on the mobility of the friendly and enemy forces, the river conditions, and the required crossing speed. [Ref. 4: p. 1-2]

Hasty river crossings are hurriedly planned, decentralized operations using existing or expedient means. Such is the case, for example, when a crossing is conducted as a continuation of an attack with little or no loss of momentum by the attacking force. In this case a hasty crossing is preferred over a deliberate crossing. Hasty crossings are feasible when enemy defenses are weak or can be overcome by our own fires, and when the maneuver forces are equipped to rapidly advance, cross, and continue the attack.

Deliberate river crossings are done when a hasty crossing is not feasible, when an earlier one has failed, or when the offensive operations commence at the river line. Plans for a deliberate crossing generally include more centralized control of crossing means, fire support, crossing time and other supporting elements than those for a hasty crossing. A deliberate crossing is required when prevailing conditions preclude the execution of a hasty crossing. This generally means that the enemy defenses are very strong or that the river obstacle is very severe and cannot be crossed by expedient means.

When a division commander has received a warning order of a river crossing operation from the higher command headquarters, he first announces to his staff about their own mission. After that the commander and his staff will work to prepare the river crossing plan by a sequence of commander and staff actions as mentioned before. In order to prepare a river crossing operation plan, the engineer

battalion commander, who is a special staff of the division commander for the engineer system, surveys the river crossing area to estimate the various elements of the river crossing operation. To accomplish his mission, he cooperates with the operation staff(G-3) and the logistic staff(G-4) to get some kind of information about the combat organization of the division for this operation and logistic data about the division.

Based on the information obtained, he first examines to see if bridges exist in the division's operation boundary. This is done because he needs to estimate the required number of crossing sites based on the information about the combat organization, and logistic data. For example, usually each division needs 6 raft sites and 2 bridge sites of class[1] 45 or more. However, if there is already one bridge of class 45, then he needs only 1 more bridge site and 6 raft sites to accomplish the river crossing operation.

After examining the existence of bridges in the specified area and estimating the required number of additional crossing sites, he selects the crossing site by using any available information about the river at each crossing site such as width, depth, velocity, bottom condition etc. If he has no information about the river at the crossing sites, estimation would be very difficult because it's hard to get this kind of information in the battle field. Therefore the engineers need to keep this kind of information with them through the terrain studies before conducting a river crossing operation.

Once the crossing sites, have been selected, the engineer battalion commander can estimate the requirement of the equipments. However if there is not enough equipment to support a division's river crossing operation, then he recommends to the higher command headquarters to get support from other engineer units. Through a sequence of this kind of action, the battalion commander and G-3(operation staff) make a river crossing plan and finally recommend the course of action to the division commander. River crossing operations are generally very difficult, because we must estimate accuratly and rapidly what is to be done for this kind of operation.

b. Obstacle and Denial Operation

The primary purpose of obstacle use is to enhance the effectiveness of friendly fires. The use of reinforcing obstacles coordinated with existing obstacles makes it possible for a commander to delay and disrupt enemy formations. A

---

[1]Class is a gross weight of wheeled vehicles or tracked vehicles in ton. The bridge class number which will permit either wheeled or tracked vehicles to cross if the vehicle class number is equal to or less than the bridge class number.

commander can also use obstacles to allow him to distribute his forces advantageously, strengthening one area with defensive obstacles so that it can be lightly defended to free the forces to be used elsewhere. He can also use obstacles in conjunction with mobile forces to protect his flanks and other lightly defended areas. In addition to these employment of obstacles, the destruction of a transportation network also delays the enemy's progress [Ref. 5: p. 3-4].

In the modern battle field, if the enemy has a stronger combat power than the friendly forces, then our defense position can be destroyed and we may have to conduct retrograde operation to establish new defense positions. As a result, a enemy may occupy certain area once occupied by friendly forces. During a retrograde operation, we try to gain more time in order to retreat safely and to prepare new defense positions. Also if we don't destroy targets such as bridges, tunnels, dams, airfields, etc., then our enemy can use these facilities to advance expeditiously. And if we don't destroy facilities of the key industries, then our enemy can use these facilities after occupying them. Thus we make denial operation plans and conduct denial operations to destroy certain targets.

But the object of a retrograde operation is to occupy a new defense position and to conduct counterattack after destroying the enemy forces at the new defense position. If we destroy all of the deny targets and emplace all the designated obstacles during a retrograde operation, there will be obstacles for our counter attack. It will then require enormous efforts and resources to overcome the destroyed targets and breaching the installed obstacles. And time is a very critical factor to conduct denial operations.

In other words when do we have to destroy some or all of the designated targets? If we have destroyed a certain bridge before our forces have crossed this bridge, or if we fail to destroy this bridge before enemy forces arrives, then a very critical problem for the next operations will occur. Therefore a commander has to make a very careful decision which target and at what time do we have to destroy a particular target.

Also obstacle and denial operation plans can be developed under circumstances of offensive operations. When we try to attack the enemy, we need to concentrate major combat power in the frontal area of a battle field. As a result of the concentration of combat power, the franks of the combat boundary might be weak against the enemy's counter attack during an offensive operation. For this reason, we

have to use obstacles to reinforce the franks, thus reducing the threat of an enemy's counter attack. Consequently, we have to develop the obstacle and denial plan not only for retrograde and defensive operation but also for offensive operations.

Consider the case that a commander needs certain amount of time to prepare a new defense position. Then he first gives a planning guidance to his staff for preparing a barrier and denial operation plan. After receiving the commander's planning guidance, his staff will survey the designated obstacles and denial targets within certain segments of the roads. Based on these information, they estimate the delay time of already emplaced obstacles and denial targets. But if the estimated delay time is not enough for preparing a new defense position, then additional targets must be selected from the previously planned targets and obstacles such as minefields, road craters, abatises, and bridges, and tunnels. At this point, his staff has to consider which types of target will be selected, because it's a very important factor for the following operation after finishing an offensive or retrograde operation. Therefore, a commander's planning guide has to contain criteria for choosing additional targets and preparing obstacle and denial plans.

However, if the total available delay time, when all types of designated obstacles are emplaced and all types of targets are denied, is not enough for the preparation of a new defense position, the commander will have to change his planning guidance. And his staff can proceed to complete a plan in accordance to the commander's new planning guidance.

c. Recovery Operation

Maintaining mobility is one of the responsibilities of the engineer units that are members of the combined arms team. A recovery operation is to recover destroyed structures or facilities by friendly or enemy forces. As mentioned in the denial operations, we will destroy certain facilities to deny their use by the enemy and also the enemy will destroy their facilities to deny being used by our friendly forces. All of these destroyed facilities will be obstacles to the movement of friendly forces when we conduct counter attack or offensive operations. Therefore, engineers have to recover these facilities.

Recovery can be divided into two categories: permanent recovery and temporary recovery. In recovery operations, the time factor is the most important one because they will be used in the next operation. So we have to maintain detail recovery plans and enough resources and equipments to complete a task within the required time.

When we make a recovery plan, we have to consider how to recover and what material will be used for. In war time, we usually do temporary recoveries because we have not enough time to perform permanent recovery completely and also we have not enough material for that.

III. SYSTEM ANALYSIS AND REQUIREMENTS

A. INTRODUCTION

As described in Chapter II, there are many problems in order to plan and conduct any kind of military operations, and the planning process of an operation plan is complicated. In this chapter, I will describe what are the problems, what is the existing system being used in the ROKA operation planning process, what are the requirements of a desired system, what are the functional aspects of a computer based operation planning support system, what must be done to solve the problem at each level of an operation, and what is the logical application of a system.

B. PROBLEM DEFINITION

In the ROKA, each level of command has different field movement exercises such as Team Spirit, which is the world's largest annual field movement exercise conducted by the combined forces of the Republic of Korea and the United States of America, division field movement exercise, command post exercise(CPX), which is an exercise for the commander and staffs in the field, and so on.

Each command has to make an operation plan for an exercise every time. Staffs at each level of the command spend a lot of time to prepare it because they need information about the area where they conduct the exercise and the area changes from exercise to exercise. As a consequence, various problems arise.

- Waste of time and effort because they repeat the same work manually.
- Data redundancy and inaccuracy because each command gets data and information separately about the same objects. As a result, the information and data are a little bit different each time for the same thing. For example, coordinates of certain point, width, depth, or velocity of river etc. are invariably different when specified by different persons. Also it is difficult to get data outside a unit's own boundary, if the operation is conducted outside its area.
- Lack of data integration and security problem because they are kept in the file book. Also it occupies a large space.
- Difficulty to update plan or data in the file book because they have to rewrite the different part of the operation plan and distribute it to the higher commands, adjacent units, and surbodinate units.

C. DESCRIPTION OF EXISTING SYSTEM

All military units need to get various kinds of data and information in order to make their own operation plan and estimate surrounding situations to prepare counter action against the enemy. As mentioned before, in current circumstances in the Korean Army, the engineer units perform two important engineer missions - river crossing operation and obstacle emplacement operation - to support the combined team. To accomplish these missions, engineer units need various relevant data and information, which they collect through terrain studies or the use of other methods. In addition, engineer units always keep data and information about terrain, roads, rivers, deny targets such as bridges, dams, airfields etc., and obstacles that are classified or designated as targets.

Once the engineers of a unit gets information about certain things or makes a plan, they keep one or more copies with them and send several copies to their higher level commands. Therefore all higher level commands such as the Army Headquarters, the Field Army command and the Corps Command keep a lot of file books for this kind of data and information. If a certain unit gets a specific mission, the working officers participating in the operation have to search first the file books to get the appropriate information, then examine the relative situations, and finally make a plan accordingly.

D. REQUIREMENTS SPECIFICATION

The analysis phase of the system development involves project planning and system requirement definition. Requirements specify the capabilities that the system must provide. These include functional requirements, performance requirements, and user interfaces, as well as development standards and quality assurance standards.

Requirement specification based on the System Definition is a technical specification for the system. The goal of requirement definition is to completely and consistently specify the technical requirements for the desired system in a concise and unambiguous manner, using formal notations when appropriate. Requirement specification states the "what" of the software product without implying "how". System design is on the other hand concerned with specifying how the system will provide the required features [Ref. 6: p. 88].

1. Applied Environment of Required System

As described in the Chapter II, the engineer system conducts various engineering operations to support the army operation as a member of the combined

forces. In order to conduct any kind of operation, each unit needs an operation plan prepared through a sequence of commander and staff actions. Our system is expected to be used to support the planning process for these actions.

If a commander receives a warning order, whether a river crossing or an obstacle/denial operation, from higher headquarters, the commander and his staff will try to get relevant information to make a suitable plan. At this point, this system can be used to support their decision making process to complete a plan more rapidly.

2. **Description of Requirements**

We can divide the desired computer system in two different parts according to the two major operations, river crossing and obstacle/denial operation, because these two operational functions are quite different from each other.

a. River Crossing Environment

In the river crossing operation environment, the computer system has to perform the following functions to support the planning;

- *Estimate Necessity*: If certain unit tries to attack the enemy across a river, the commander and his staff have to decide whether a river crossing operation is needed or not. A river crossing operation may not be needed, if there are available bridges in the operational boundary. Therefore, our system has to have a function estimating the necessity of river crossing operation in a situation.
- *Estimate Possibility*: If a river crossing operation is required, the commander and his staff will estimate the possibility of an operation based on the availability of their own engineer unit and equipment. Therefore, our system must do this kind of function.
- *Select Crossing Sites*: The system is required to perform the selection of the required number of crossing sites and to provide the information about each crossing sites including width, depth, velocity, and bottom condition of the river because these are needed to make a river crossing plan.
- *Calculate Equipments and Assemble Time*: After selecting the crossing sites, the system must calculate what kind of and how many equipments are needed to carry out the operation. How much time is needed to assemble these equipments should also be provided.
- *Estimate Available Supporting Unit*: If available equipments are not enough for a specified operation, then additional equipment must be sought. Therefore, our system has to have a function to search which units have equipments available and how many equipments can be used to support this operation.
- *Estimate Crossing Rate*: Finally, a system must generate the final output that contains the number of vehicles or equipments that can cross the river every hour from the start of the operation(H-hour) and an estimate of the time when

all vehicles and equipments have crossed the river completely. The commander and staff can make a decision and a detailed plan based on this final output.

b. Obstacle/Denial Environment

In obstacle/denial operation environment, a desired system has to perform the following functions to support planning;

- *Estimate Necessity*: During retrograde operation, a commander establishes obstacles to gain certain amount of time in order to withdraw safely and to prepare new defense positions. For example, if the commander needs 5 hours for this purpose, and some obstacles are already emplaced, then he has to estimate at this time either he needs more obstacles to secure 5 hours, or he doesn't need any more if he already has secured more than 5 hours through the already emplaced obstacles. Our system must have this kind of function to support the decision making.
- *Select Targets*: In case the already secured time is less than 5 hours, the commander will emplace additional obstacles to secure the needed 5 hours. He must survey and select some or all the designated but unemplaced obstacles and enable them. So our system has to perform this function, too.
- *Extend Segment*: If all obstacles are emplaced in a certain segment of a road, and the available delay time is still less than 5 hours, the commander has to extend the segment further to secure 5 hours. Our system must perform this operation as well.
- *Characteristics of Targets*: Frequently we need to know the characteristics of certain type of targets. Our system must provide us this information as needed.

E. ANALYSIS

The objective of analysis is to determine what must be done to solve a specific problem. The basic function of any data processing application is to convert the input data into the desired output information.

In this section, I shall attempt to develop a complete functional understanding of the proposed system. The intent is not to determine how the system will work, but what it must do to solve the problem. As mentioned before, this thesis, we shall concentrate on the two different operational aspects, river crossing and obstacle/denial operations. [Ref. 7: p. 49]

1. River Crossing Environment

River crossing is a special operation, requiring detail planning and information. To solve this problem, first I will use the high level data flow diagram(DFD) to define logically the proposed system, and later expand the high level DFD to develop the functional level DFD to define the functional application of our proposed system. [Ref. 7: p. 55]

A DFD shows data sources and destinations, data flows, data stores, and processes. A DFD uses five basic symbols to form a picture of a logical system in this thesis. A square defines a source or destination of data. An ellipse represents a process that transforms data. An open-ended rectangle is a data store. A rectangle represents a database. And an arrow is used to identify a data flow.

a. Define the Logical Model

In order to solve the previous mentioned problem and satisfy the requirement specification, we need to know who will cross the river and which area of the river will be used for this operation. Based on this initial information, the problem definition, and a sequence of commander and staff actions, a high level DFD can be developed as a logical model of our system for river crossing operation as shown in Figure 3.1.

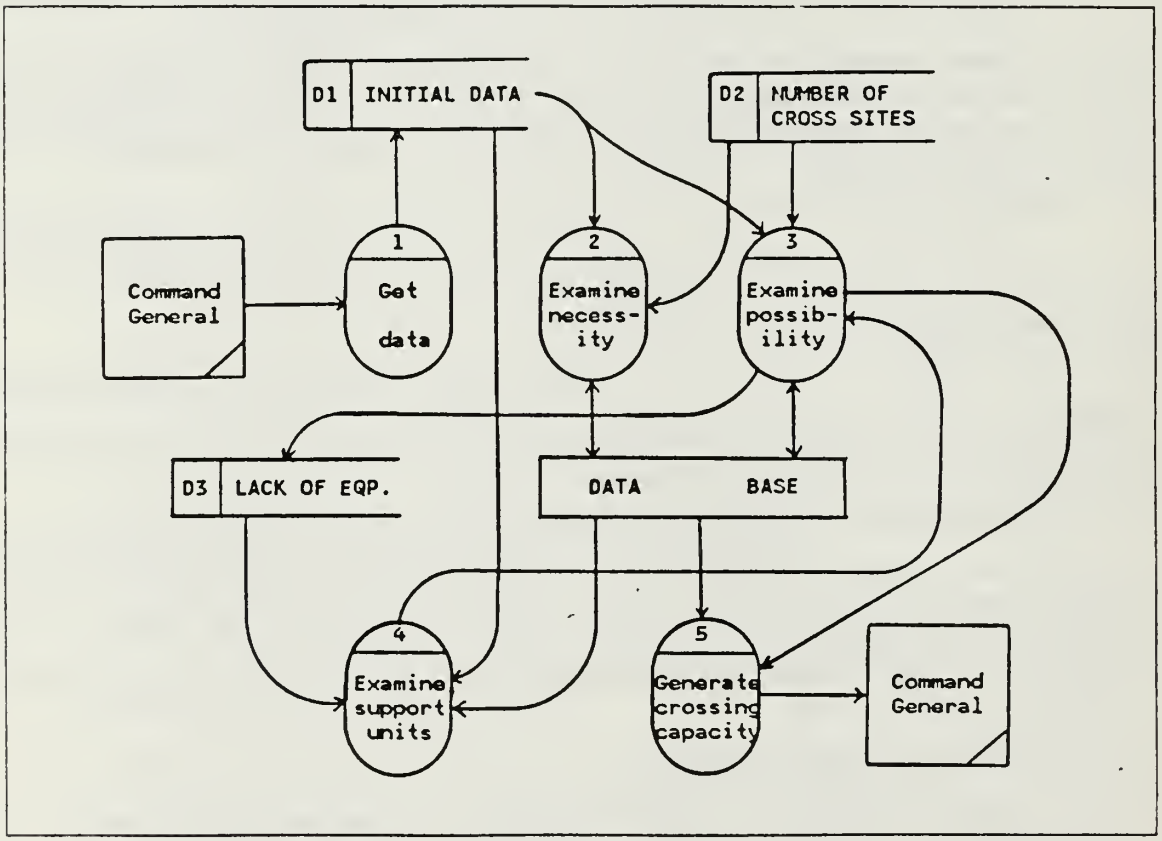

Figure 3.1 High Level DFD for River Crossing Operation.

In the first step of this DFD, our system determines whether a river crossing operation is necessary. If the output of this process is that the river crossing operation is necessary, then the system estimates whether is it possible to conduct the operation by using the command's engineer units and their equipments. If possible, then the system calculates the the vehicle crossing rate. If either insufficient manpower or equipment is found, the system estimates which units are available to support this operation and what equipments are available. After completing the high level DFD, we have to examine if the developed DFD can solve the problems or not. If positive, then we expand the DFD to the next level of details; otherwise redefine the DFD to try again.

b. Develop the functional level DFD

Expanding the data flow diagram through functional decomposition is breaking down functions into its subfunctions. These low level functions then become processes on a new data flow diagram, complete with their own data stores and data flows. [Ref. 7: p. 56]

(1) *Functional decomposition of Level 2: Estimate Necessity.* In order to estimate the necessity of river crossing operation in this level, the system, as said earlier, examines if there exist available bridges within the operational boundary. The system can also estimate the capacities of these bridges based on the stored data. The result of the calculated number of crossing sites is stored for other processes to use later. Thus the functional decomposition at this level is as shown in Figure 3.2.

(2) *Functional decomposition of Level 3: Estimate Possibility.* At this level, the system has to estimate the possibility of a river crossing operation, using the command's engineer units and their equipments. In order to accomplish this function, first the system needs to examine the organized engineer units and their equipments as an output of this process. Also the system has to select the required crossing sites based on the previous process. After selecting the crossing sites, the required number of equipment sets have to be calculated as in the process 3-5. At this point, the required floating bridges can be calculated, but the equipments of each raft site can not be calculated because system does not know how many rafts are needed at each raft site, as the width is not yet known. For example, we need one raft per site up to 100m of river width, two rafts per site up to 200m, and three rafts per site up to 300m at each crossing site. Therefore we need one more subfunction to determine the number of raft for each raft site before calculating the required amount of equipments for raft sites.

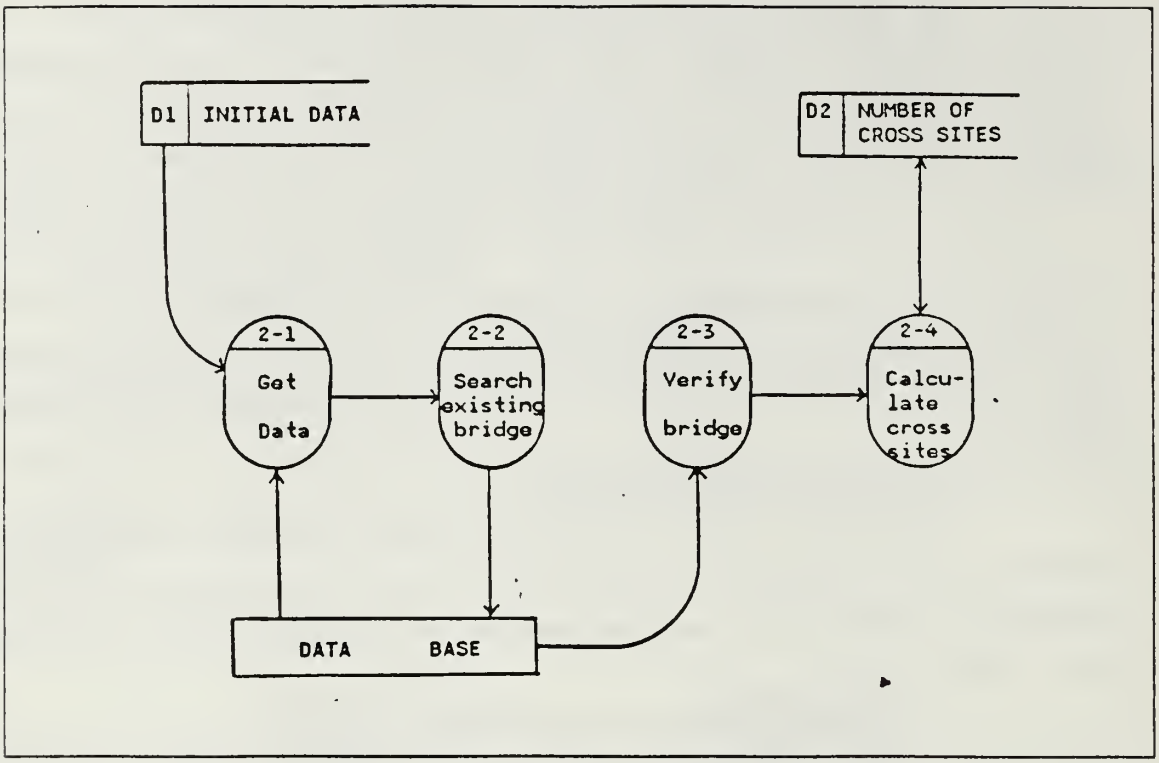

Figure 3.2 Decomposed DFD of Estimate Necessity Process.

Finally, the system can estimate the possibility in process 3-6 of Figure 3.3 by comparing the required equipment sets in process 3-5 and the available equipment in process 3-2. At this time, if the available equipment is enough to satisfy the required amount of equipment calculated by process 3-5, it means that it is possible to conduct the river crossing operation without additional support coming from other engineer units at higher headquarters. But if it is not enough, then the commander has to make a final decision of what to do. The entire process is functionally decomposed as shown in Figure 3.3.

(3) *Functional decomposition of Level 4: Examine supporting Units.* If the commander decides that he needs support from higher headquarters for his operation, the system proceeds to find available engineer units and their equipments. It then reestimates the feasibility of accomplishing the operation under the new circumstances. With the new result from process 4-3 of Figure 3.4 the commander can then make a recommendation to his higher headquarters where a final decision is to be made. The functional decomposition of this process is shown in Figure 3.4.

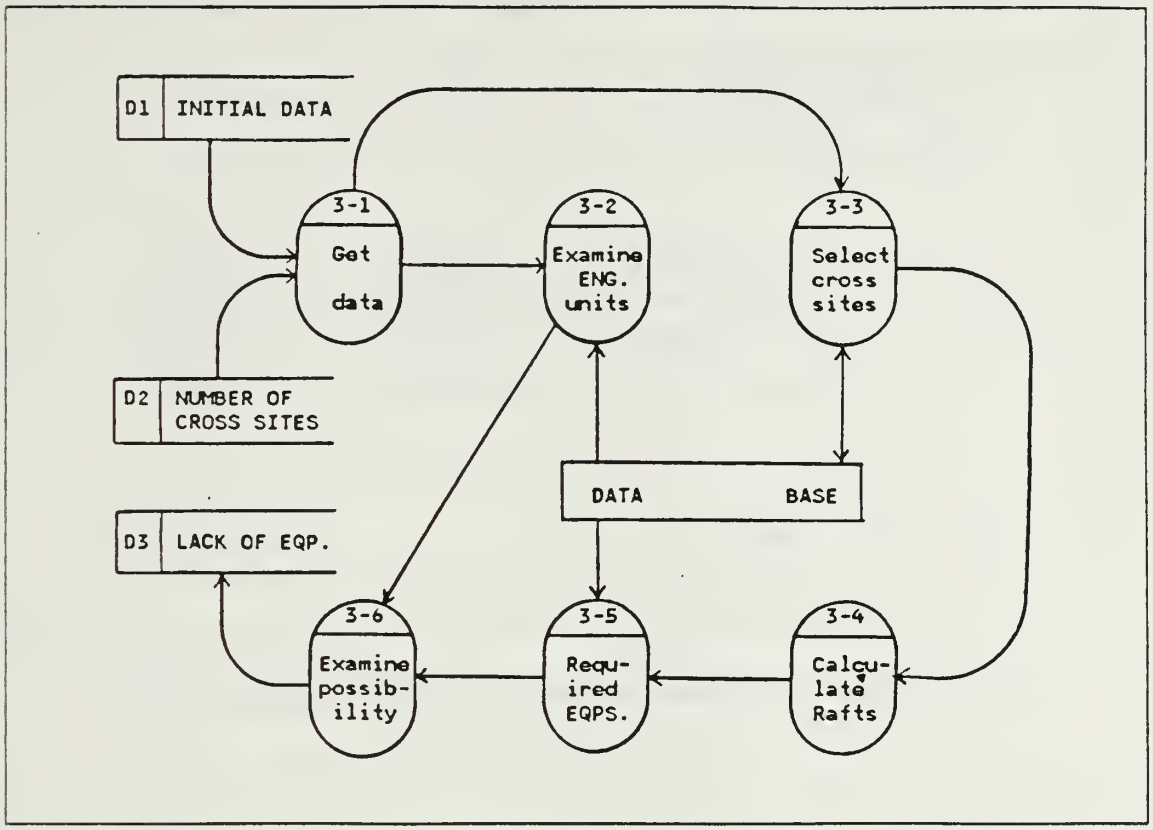

Figure 3.3 Decomposed DFD of Estimate Possibility Process.

(4) *Functional decomposition of Level 5: Generate Crossing Capacity.* From the result of the above 4 functional levels, the final feasible plan of the river crossing operation is generated. The system next begins to estimate the number of trips necessary at each raft site (process 5-2 of Figure 3.5). It also calculates the assembly time for each crossing site, the transportation capacity for vehicles and combat equipment such as tanks, armored vehicles etc. and the time, called the H-hour, when everything has crossed the river, Figure 3.5 shows the functionally decomposed DFD of this level.

c. Develop the Data Dictionary and Algorithms

Given a complete data flow diagram, we have a reference document specifying the entire decision support system, we can now begin to consider the details such as the data elements and the algorithms.

To allow us to have a better understanding about data, we make use of a data dictionary. A data dictionary is a collection of data about data. The basic idea is

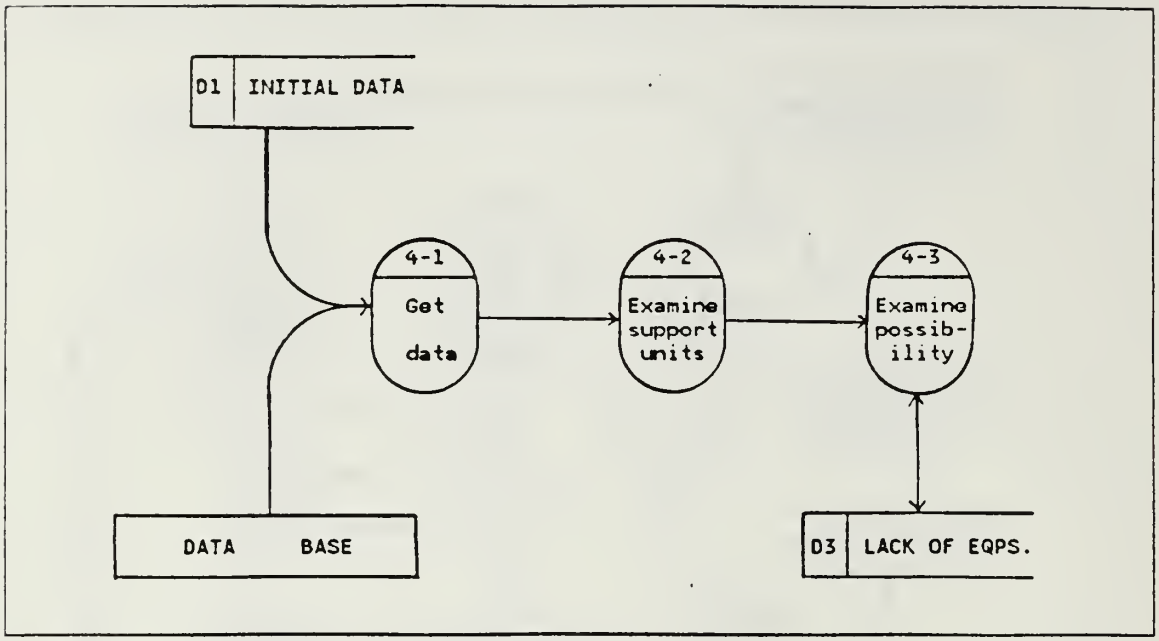

Figure 3.4 Decomposed DFD of Examine Supporting Units.

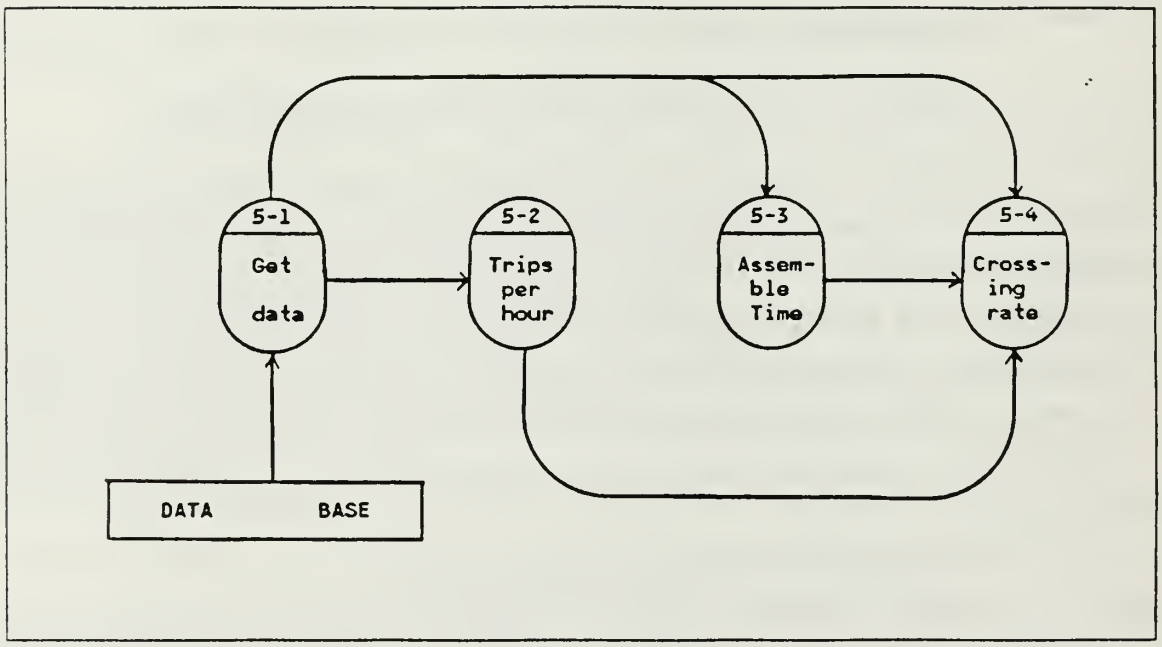

Figure 3.5 Decomposed DFD of Generate Crossing Capacity.

to provide information on the definition, structure, and use of each data element in a system. A data element is a unit of data that can not be decomposed. For each data

element, such information as the element name, description, format, source and use is recorded. The data dictionary helps the analyst to organize information about data, and is an excellent tool for communicating with the user. Additionally, the data dictionary serves as a memory aid. Key information must be recorded for each data element [Ref. 7: p. 202].

A simulated data dictionary, introduced by Davis' book [Ref. 7: p. 297], is to be used in this thesis. To maintain a semblance of direct access, information related to each data element will be recorded on a separate 3x5 filing card and a minimum amount of information will be recorded for each data element as Table 1.

TABLE 1
SAMPLE DATA DICTIONARY

```
NAME : BRGSITE
DESCRIPTION : The required number of bridge sites
              in the operational boundary.
FORMAT : Numeric ; 1 digit
LOCATION : Output to display and printer
```

In very general term, every computer system converts input data into output information, and the algorithms define the rules for these conversions. Without a general sense of the algorithms for each process or function, we can't have a good idea of what is being done in the processes in the system. The preliminary algorithm descriptions recorded on a IPO(Input/Process/Output) chart as shown in Figure 3.6 give us a general structure that can be refined later during the detailed design phase.

2. **Obstacle/Denial Environment**

An obstacle/Denial operation is also a major mission of engineer system. If we conduct an obstacle/denial operation during a retrograde or defensive operation, the commander has to consider the whole operation carefully. Ineffective planning of this operation can cause severe problems if a counter attack is to be conducted subsequently. To solve this problem, we first develop a high level data flow diagram to define the logical model of the proposed system. After that, we develop the functional level DFD to define the functional application of the proposed system by expanding the high level DFD into more details. Finally, we develop the data dictionary and algorithms for this system. [Ref. 7: p. 55]

IPO CHART

SYSTEM : River Crossing Operation Support System
PREPARED BY : Jang, Chang Ki
MODULE : CALCSITE ALGORITHM DATE : APR/87

CALLED OR INVOKED BY:

EXISTBRG

CALLS OR INVOKES:

INPUTS:

AREA (coordinate)

Outputs:

A_BRG, A_RAFT
B_BRG, B_RAFT
BRGSITE,RAFTSITE

PROCESS:

```
WHILE NOT EOF DO
   A_RAFT = A_RAFT - 1, B_RAFT = B_RAFT - 1
   A_BRG  = A_BRG - 1,  B_BRG  = B_BRG - 1
END WHILE
BRGSITE  = A_BRG  + B_BRG
RAFTSITE = A_RAFT + B_RAFT
```

LOCAL DATA ELEMENTS:

ANEED, BNEED

NOTES:

Figure 3.6 An IPO Chart as an Example.

a. Define the Logical Model

As previously mentioned in the system requirement specification section, the system can be defined logically as shown in Figure 3.7. On the first level of this diagram, the system gets its initial data from a user about the segment of a road represented by the coordinates indicating the start and end points, the road number that we want to conduct the operation, and the required delay time to the enemy.

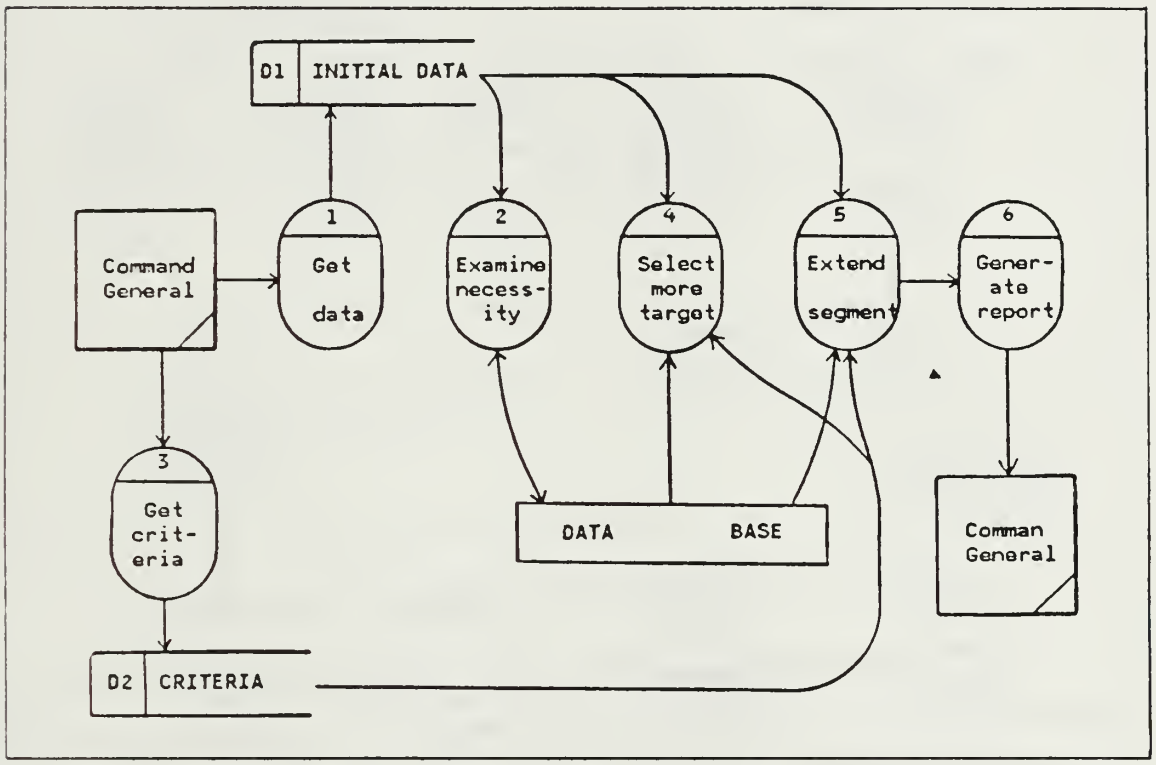

Figure 3.7 High Level DFD for Obstacle/Denial Operation.

At the second level of the DFD, the system estimates what obstacle/denial operations are required to get the required amount of delay time. If more obstacles are necessary, the system selects the additional targets to satisfy the commander's requirement. In order to select additional targets in this process, the system needs some additional data and the selection criteria as contained in the commander's planning guidance. More detail about this selection criteria will be described in the next phase of developing the functional level DFD. After selecting the additional target, the system evaluates again the whole process. However, it may not be enough

when all the designated targets in this segment have been selected. In this case, we go to process 5 to extend the road segment to find a solution.

b. Develop the functional level DFD

At this point, the high level data flow diagram is expanded as done in the river crossing operation through functional decomposition.

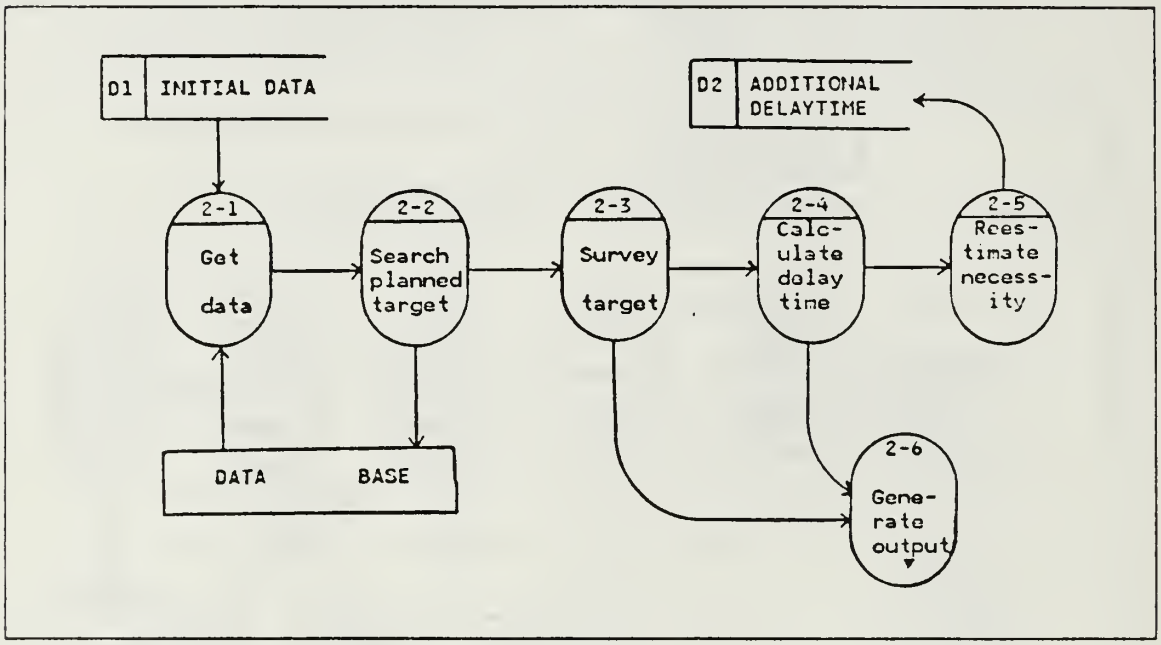

Figure 3.8 Decomposed DFD of Estimate Necessity Process.

(1) *Functional decomposition of Level 2: Estimate Necessity.* In order to estimate the necessity of an obstacle/denial operation, we first need to get data from the user and the database. After getting the data, the system proceeds to search the designated targets in this segment. As seen in Figure 3.8, in process 2-2, the system verifies the intended targets and calculate the delay time.

In the final process at this level, the system evaluates the whole operation based on the calculated delay time and the commander's required delay time. If the calculated delay time is not enough for satisfy the requirement, we have to prepare additional operation to satisfy it. Thus, the high level DFD can be functionally decomposed as shown in Figure 3.8.

(2) *Functional decomposition of Level 3: Select Target.* As mentioned before, at this stage, the system selects obstacle targets that will be denied or emplaced

based on the commander's selection criteria. Generally the criteria can be divided into two parts: major criterion and minor criterion. Major criterion can be either minimum number of targets or maximum number of targets, and minor criterion can be either maximum recover time or minimum recover time.

TABLE 2
SAMPLE DATA FOR SELECTING TARGET PROCESS

TNUM	TYPE	DELAY TIME	RECOVER TIME
A	BRG	1.5 H	10 H
B	BRG	1.5 H	9 H
C	BRG	0.6 H	4 H
D	BRG	0.5 H	5 H

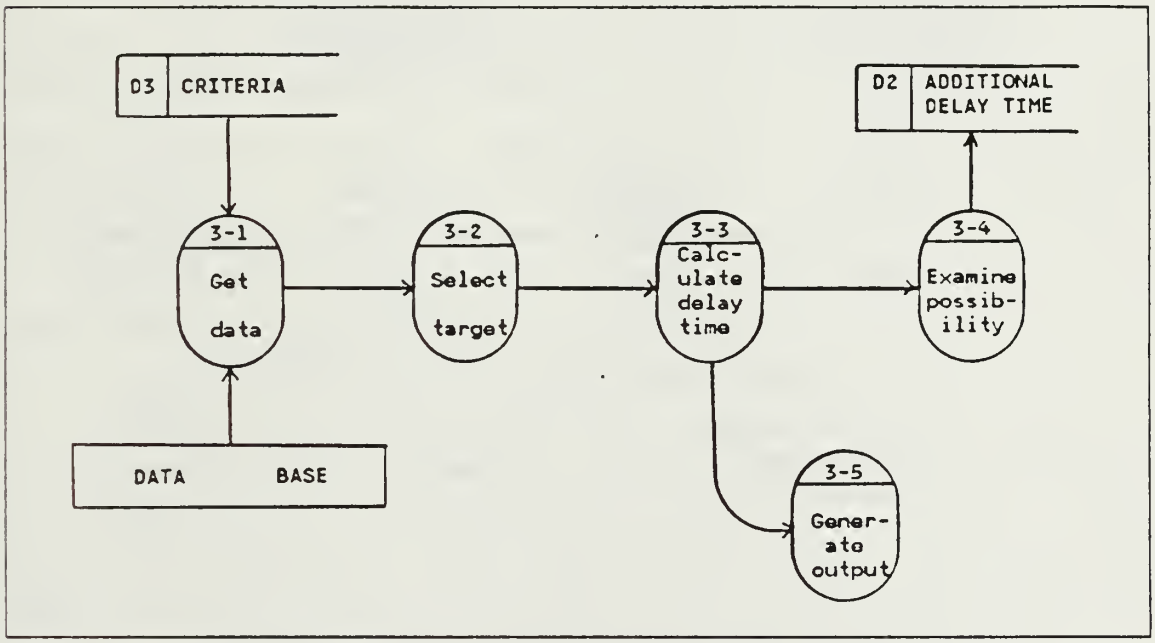

Figure 3.9 Decomposed DFD of Select Target Process.

For example, when the commander has selected minimum number of targets for the major criterion and minimum recover time for the minor criterion, the

targets' delay and recover times are as shown in Table 2, and the commander's required delay time is 2 hours, the system will select the target B and C in process 3-2 of Figure 3.9. It can easily be verified that this combination indeed satisfies all the stated conditions. Functions for other processes are as shown in Figure 3.9.

(3) *Functional decomposition of Level 4: Extend Segment.* If it's not possible to satisfy the commander's requirement when all designated targets in this segment have been selected, we have to extend the segment beyond the end points to find other targets. The result of this, if based on minimum possible extension of the segment will be recommended to the commander. In order to do this kind of processing, data is needed from the initialization process to determine the direction of the road that allows an extention since the other direction is presumably occupied by the enemy forces. After determining the direction of road, the system can then search for targets to satisfy the requirement. Figure 3.10 shows this process and others as well.

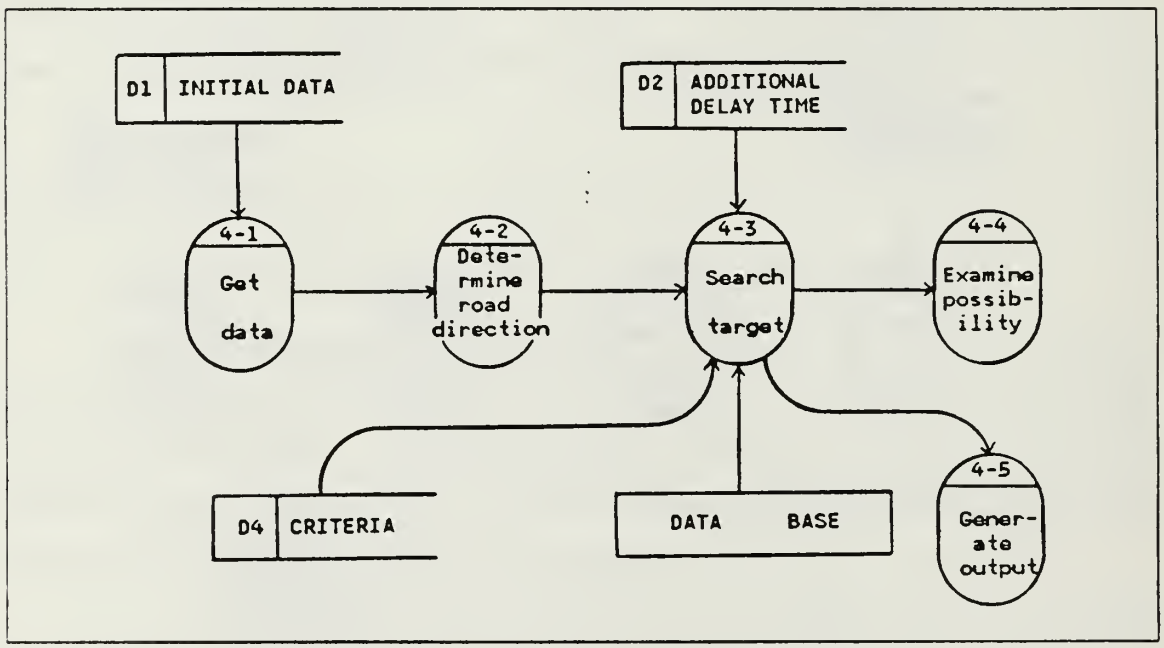

Figure 3.10 Decomposed DFD of Extend Segment Process.

IV. SYSTEM DESIGN AND IMPLEMENTATION

A. OVERVIEW

The proposed system that I try to develop in this thesis is a Decision Dupport Dystem(DSS). "The term decision support system refers to a class of systems which support the process of making decisions. Decision support system allows the decision maker to retrieve data and test alternative solutions during the process of problem solving" [Ref. 8: p. 368]. How are decisions made? The answer affects the design of computer based information systems to support the decision making process.

In the previous chapter, analysis has just been completed. The functional requirements of the desired system have been documented with a data flow diagram, a data dictionary, and a set of algorithm descriptions. The time has come to consider how to do it, and thus the system design process begins.

"The objective of system design is to produce an appropriate physical model of the system, called system flowchart, within the context of a complete system and to determine how the system will be implemented. In the structured approach to systems analysis and design, we begin by constructing the logical model, often using data flow diagram as a tool. During system design, this logical model must be converted to physical form" [Ref. 7: p. 326]. At first, we develop an appropriate system flowchart that represents the overall system, and the database, which is one component of the overall system, is designed. Finally, we will develop a set of specification for those programs using Hierarchy plus Input Process Output(HIPO) technique.

During the program design process, if we can determine what our minds do at a given stage of any decision making process, we can easily find in this program design a section that corresponds to an equivalent aspect of human intelligence. All the elements that comprise the human decision making process - goals, facts, rules, inference mechanism, and pruning - must be collected in a computer program for it to be properly described as possessing Artificial Intelligence(AI). [Ref. 10: p. 10]

B. DEVELOP THE SYSTEM FLOWCHART

In this Section, I will identify the individual physical components of the system. Contents of these black box will be planned in the next section--database design and program design phase.

1. **Physical Components of the System**

In order to develop the system flowchart, we first need to identify the physical components of the system as shown in Table 3. The proposed system will access the database through the Data Base Management System(DBMS) and also will need initial data to process each operation. It will write a variety of reports phase by phase of each operation.

TABLE 3
PHYSICAL COMPONENTS OF THE SYSTEM

Programs	Manual procedures
Data entry	Data collection
Initialization	Making plan
Intelligent System	
River crossing operation	
Obstacle/Denial operation	

Software	Files	Forms
DBMS	Data Base	Reports

A data entry program will maintain the database: to add data to the database, to update data which is in the database, and to delete data from the database. A data initialization program will initialize the operation environment for each operation, and will store it in the on-line storage for later use. River crossing program and obstacle/denial program are the intelligent part of the proposed system that will generate the report based on the available data in the system. AI techniques allow the construction of a program in which each piece of the program represents a highly independent and identifiable step toward the solution of a problem or set of problems.

2. **System Flowchart**

A system flowchart is a traditional tool for describing a physical system. The basic idea is to provide a symbol to represent, at a black box level, each discrete component in the system - programs, files, forms, procedures, and so on.

A data flow diagram presents a rather abstract picture of the system. In contrast, the system flowchart is more concrete. Specific programs or procedures replace the generalized process, and specific files or databases replace the data stores. Given the flowchart, it is possible to visualize how the system will be implemented. The system flow diagram identifies each of the discrete components of the system. For planning purpose, these components can be attacked one at a time - the divide and conquer approach. [Ref. 12: p. 87]

The system flowchart of the proposed system graphically illustrates the physical system. As shown in Figure 4.1, each symbol defines at the black box level one of the discrete components that make up this system, and the flowlines define the logical path through the system. Thus the system flowchart as shown in Figure 4.1 is generated, based on the physical components of the system and data flow diagram developed in the previous chapter.

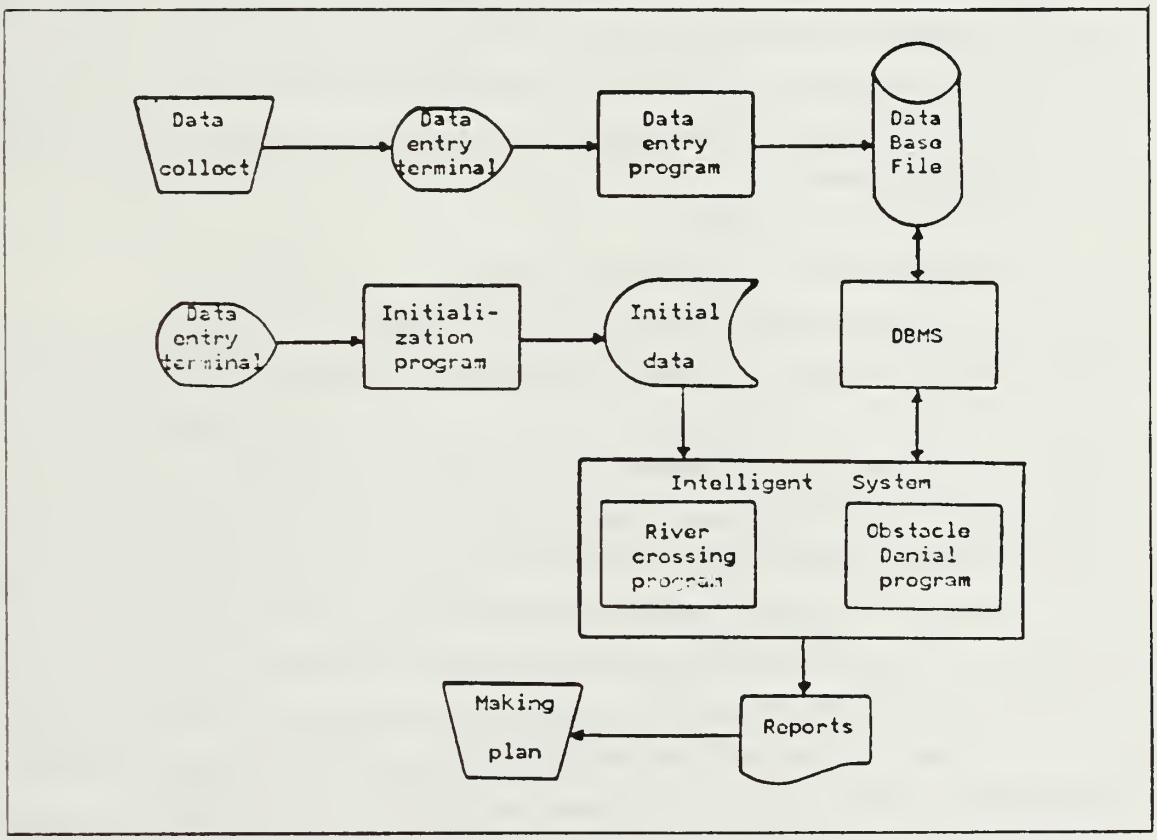

Figure 4.1 System Flowchart.

C. DATABASE DESIGN

1. Overview

As I mentioned before, the proposed system has a database as a part of the physical components of the system. In the proposed system, the database is maintained with data to satisfy the applications for each operational environment, namely river crossing operation and obstacle/denial operation.

In order to accomplish our goal, two major tasks are accomplished during the design of a database-oriented system. One task is to define the database structure, and the other is to design the application programs. Designing the application programs will be discussed in the next section, program design.

Database design is an intuitive process. Typically, it is an iterative process. During each iteration, the goal is to get closer to an acceptable design. Thus a design is developed and then reviewed. Furthermore, the design process is divided into two phases. First, the logical structure of the database that is a DBMS-independent database design is developed. Once the logical structure has been defined, it is transformed into physical form that conforms to the features of the DBMS to be used. In this way the logical structure is mapped into the data description facilities provided by the DBMS product.

2. Logical Database Design

a. Procedures for Logical Database Design

The major steps in the logical design process are divided into five phases. First, the data dictionary is processed and the data to be stored is identified and segregated. Next, the terms used for the data are to be clarified because there may be hundreds or thousands of data items in the logical schema which is developed in the next phase. The third step in the design process is to develop the logical schema by defining records and relationships. Records are defined by determining the data items they will contain. And the relationships between these records is to be determined in this phase. The next step in logical database design is to define the processing of the database. But this has already been defined in the system analysis phase. Here, however, it will be defined in more detail for the program design phase. Finally, the logical schema and user views are examined in light of the requirements and program descriptions. To be effective, the design review must follow stringent guidelines. [Ref. 13: p. 181]

From the result of these processes, the logical database records and relationships among them are generated as an output of the logical database design process as expressed in the form of data flow diagrams and policy statements, and the data dictionary will become the input to the logical database design.

b. Develop the Logical Database Records and Relationships

A logical database design specifies the logical format of the database. The records to be identified and maintained, their contents, and their relationships are specified.

(1) *Logical database records.* In this section, logical database records are identified to satisfy the required operation and the contents of these records are defined. The proposed system has 11 data records to accomplish the required functions, as shown in Table 4. The contents of these records, names of fields, and their formats are specified. For example, River Object(RIV_OBJ) record contains data about all objects such as bridges and river crossing sites of a certain river. So RIV_OBJ record should contain the name of river, location of the objects and type of object etc. Also, available equipment of each unit(UNITEQP) record represents how many river crossing equipments are being equiped in each engineer unit. So it contains the information about the unit name, the amount of equipments that are available for the operation, and so on.

(2) *Relationship between records.* As stated several times already, the essence of a database is the representation of record relationships. These relationships are specified during logical design. Figure 4.2 shows relationships between previously defined data records.

As shown in Figure 4.2, several river crossing means can be operated at the same crossing site and also different river crossing means can be operated at the several crossing sites simultaneously. Therefore the relationship between crossing sites and river crossing means is many-to-many relationship [Ref. 13: p. 121]. Moreover, there may not be any relationship between BRIDGES and TARGETS because some bridges can not be a target to deny, if there exist by-passes around the bridge or if it is easy to overcome, even when a bridge is blown up. Since a bridge can be on only one road or one river, there exists a one-to-one relationship between BRIDGES and RIV_OBJ or RD_OBJ.

TABLE 4
LOGICAL DATABASE RECORDS

RECORDS NAME	DESCRIPTION OF RECORDS
RIV_OBJ	All objects on the river such as bridges and river crossing sites.
RD_OBJ	All objects on the road such as tunnels, bridges, minefields, and road craters.
TARGETS	Targets that are planned to deny or emplace.
BRIDGES	Characteristics of bridges.
TUNNELS	Characteristics of tunnels.
MINEFLDS	Mine fields that are that are planned.
RDCRATER	Planned road crater on the road.
CROSSITE	Characteristics of the river.
CROSSEQP	Characteristics of the river crossing equipments.
UNITEQP	Available equipment of engineer units.
ASSIGN	Status of unit assignment.

3. **Physical Database Design**

a. Physical Database Design Process

The second stage of database design, called physical design, is a stage of transformation. The logical schema is transformed into the particular data constructs that are available for the DBMS to use. Whereas the logical design is DBMS-independent, the physical design is very much DBMS-dependent.

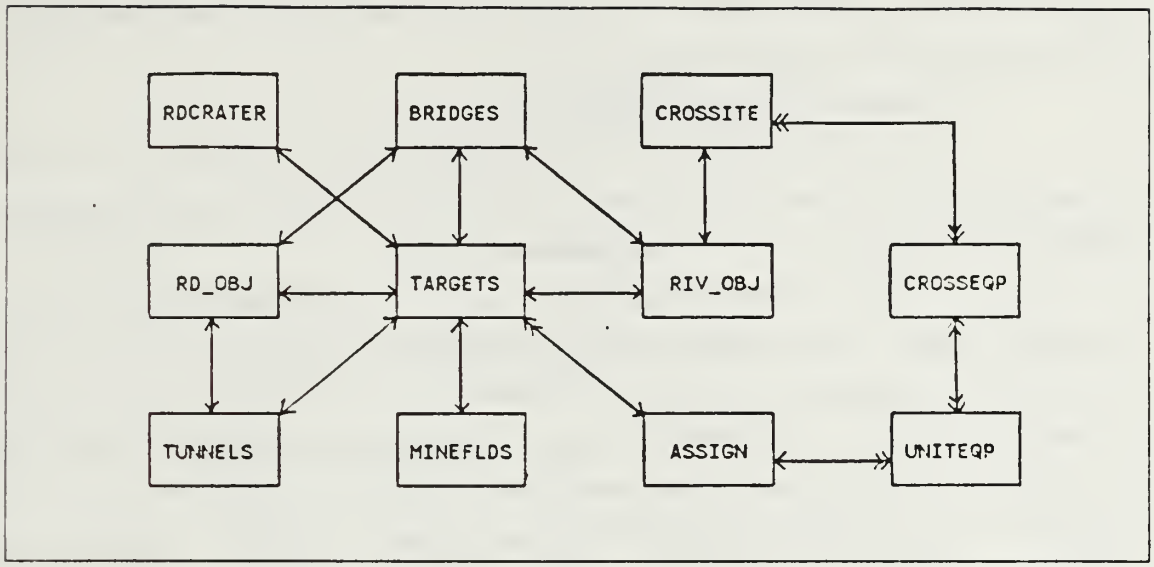

Figure 4.2 Data Structure Diagram.

A database management system is a group or collection of programs that gives the user access to a collection of information stored as data. In some type of database systems, a user writes the application programs in BASIC, C, Pascal, FORTRAN, or another high-level language. In other database systems such as dBASE III Plus, the languages are part of the database management systems, and no additional programming language is needed [Ref. 11: p. 3].

The proposed system is designed to be used in the division commands or corps commands. During any kind of army operation, division and the corps commands frequently move their command posts according to the operational situation of each phase. Currently each division and corps commands of the Republic of Korea Army already are equipped with micro-computers for the limited use of personnel management and logistic management.

From this the proposed system can conveniently be implemented as a micro-computer based system. Therefore, I think that dBASE III Plus is a good selection for this system and I so select it.

During the physical database design, two major specifications are produced as an output of the physical database design process. First, the physical specification of the logical schema called physical schema is defined. This schema is a transformation of the logical schema into the data modeling constructs available with a particular DBMS. Next, user views are defined. Since most users will need to view

only a portion of the database, the logical design must specify what user groups will view which portions of the database.

b. Develop the Physical Schema

(1) *Define the Fields and Key.* In this process, I will define the contents of each record, the name and format of each record field, the domains and the constraints, and also the keys of the database records.

Let us look at database records RIV_OBJ and RD_OBJ that are already defined in the logical database design phase. These records represent bridges and river-crossing-sites on the river or obstacles such as bridges, and tunnels on the road that have been designated as targets. In the record TARGETS, TNUM(Target Number) or LOC are keys for this record, because TNUM will identify a target. LOC is a key when we find targets within certain segment of a road. Therefore records, contents of records can be defined as shown in Table 5.

(2) *Define the Field Format and Domain.* Next, we define the format and domain of field value as shown in Table 6. In this table, LOC is represented by 2 alphabetic characters at the begining of the field value followed by 6 numeric digits. Two alphabetic characters represent the name of a coordinate block. For example, 10-XXX-11 of TNUM represents the 11th target of the 10th Corps. So "XXX" represents the corps command and "XX" represents the division command. The ARMY, CORPS, DIV, BRGD, EBN, BCO, FBCO of UNIT represent field army, corps, division, brigade, engineer battalion, bridge company, and floating bridge company respectively. Furthermore, 1:2:3 of minefield density represents the density of antitank mine, fragment antiperson mine, and blast antiperson mine respectively.

(3) *Define the Constraints.* As the requirements are evaluated and the design process progresses, constraints on data items will be identified. These constraints are limitations on the values that the data in the database can have. Three type of constraints are common. Field constraints limit the value that a given data item can have. Intrarecord constraints limit values between fields within a given record. And interrecord constraints limit values between fields in different records [Ref. 13: p. 179]. Table 7 shows an example for interrecord constraints.

D. PROGRAM DESIGN

At this time the system flowcharts have just been completed and the physical components making up the system - programs, files, forms, manual procedures, and software - have been identified. How, specifically, should the system be implemented

TABLE 5
CONTENTS OF RECORD

RECORDS	FIELDS		KEY
RIV_OBJ	LOC: Location OTYPE: Object type	RNAME: River name	LCC
RD_OBJ	LOC: Location OTYPE: Object type	RNUM: Road number	LOC
TARGETS	TNUM: Target number TTYPE: Target type STATE: Target state	LOC: Location DTIME: Delay time	TNUM LOC
BRIDGES	LOC: location LENGTH: Brg. length CLASS: Brg. capacity	BNAME: Bridge name WIDTH: Brg. width STATE: Brg. state	LOC
TUNNELS	LOC: Location WIDTH: Tunn. width	LENGTH: Tunn. length HEIGHT: Tunn. height	LOC
MINEFLDS	TNUM: Target number FRONT: Front length DEPTH: Length from front to rear	DENSITY: Density of mine field	TNUM
RDCRATER	TNUM: Target number DEPTH: Front to rear	WIDTH: Crater width	TNUM
CROSSITE	LOC: Location DEPTH: River depth OBST: Water obstacles	WIDTH: River width VELO: Water velocity	LOC
CROSSEQP	MEANS: Type of EQP. CLASS: Capacity NRAFT: Rafts/set	LENGTH: 1 set length ATIME: Assemble time	MEANS
UNITEQP	UNIT: Name of unit LTR: Light raft	AFB: Foot bridge M4T6: Float bridge	UNIT
ASSIGN	UNIT: Name of unit DATE: Attached date	BELONG: Belong unit	UNIT

based on the system flowchart, physical components of the system, and data flow

TABLE 6
FIELD FORMAT AND DOMAIN

FIELD NAME	FORMAT	DOMAIN
RIV_OBJ.LOC	CT123456	2 characters + 6 digits
TARGET.TNUM	10-XXX-11	XXX or XX or X
ASSIGN.UNIT	20 DIV	ARMY, CORPS, DIV, BRGD EBN, BCO, FBCO
RIV_OBJ.OTYPE	CHAR(4)	BRGE, SITE
MINEFLDS.DENSITY	1:2:3	Positive integer
RD_OBJ.OTYPE	CHAR(4)	BRGE, TUNN, OBST
TARGET.TTYPE	CHAR(4)	BRGE, MINE, TUNN, RDCR

TABLE 7
INTERRECORD CONSTRAINTS

RIV_OBJ.LOC	subset of	BRIDGES.LOC CROSSITE.LOC TARGETS.LOC
RD_OBJ.LOC	subset of	TARGETS.LOC BRIDGES.LOC TUNNELS.LOC
TARGETS.TNUM	subset of	MINEFLDS.TNUM RDCRATER.TNUM
UNITEQP.UNIT	subset of	ASSIGN.UNIT

diagrams? In this section, specific physical components are identified, and their interrelationships defined using the HIPO technique.

The first step in designing the program is to draw a high-level hierarchy chart. As we already discussed in Chapter III, the proposed system has two operational environments - river crossing and obstacle/denial operations - and also needs database

management functions to add, delete, and update the data in the database. Thus this system can be divided into two subsystems, river crossing and obstacle/denial, and the database management system will support these two subsystems.

1. **Subsystem 1 : River Crossing Operation**

. Based on the data flow diagram in Figure 3.1, a high-level hierarchy chart showing the primary functions of the system for the river crossing operation is defined as shown in Figure 4.3.

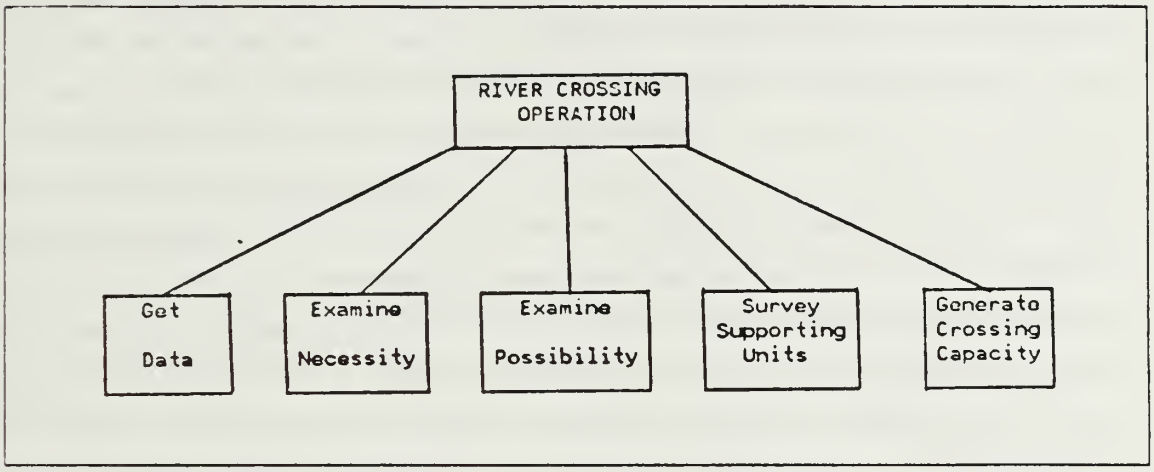

Figure 4.3 High-level Hierarchy Chart of Subsystem 1.

This subsystem has five modules, Get Data, Examine Necessity, Examine Possibility, Survey the Supporting Units and Generate the Vehicle Crossing Capacity. The River Crossing Operation subsystem begins with Process Get Data. By executing this process, the system gets the initial data about the name of the river crossing command, the name of the river to cross, and the river crossing area as represented by the coordinates. After executing this process, it returns the control to the main control module, river crossing system. Next, the main control module transfers the control to the Examine Necessity module which estimates the necessity of the river crossing operation. Then it is back to the main module, which invokes the next lower level module and so on.

By itself, the initial hierarchy chart is not very useful because it is very general. Therefore, we need to break down each function to a lower level of detail. As mentioned before, the process is known as functional decomposition. [Ref. 7: p. 329]

a. Module 1 : Examine Necessity

This module determines whether the river crossing operation is necessary or not. To perform this function, it invokes the Search Bridge function. First this function performs a "join" operation with RIV_OBJ and BRIDGES relations and finds the available bridges in this area with the given initial data. In order to find existing bridges in this area, the module determines the direction of the river. The coordinates are two dimensional value, but in the system we need only X-axis or Y-axis value to find the bridges or crossing sites. After finding the available bridges, the module invokes the Verify Bridge function to check the class of bridge and to estimate the capability of that bridge. It then calculates the required number of crossing sites based on this result. If there is no bridge available in this area, then we need to make one bridge site and three raft sites in each area. But if there exist usable bridges and the class of the bridges is over 45, then we don't have to make a bridge because heavy equipment such as tank can cross through this bridge. But if the class is between 20 and 45, then the system assumes that the capability of each bridge in this class is two times that of the rafts. Therefore, the required number of raft site will be reduced by one for each bridge in this class. Finally, the module estimates the necessity of the operation based on the calculated crossing sites. Therefore we can decompose this module as shown in Figure 4.4.

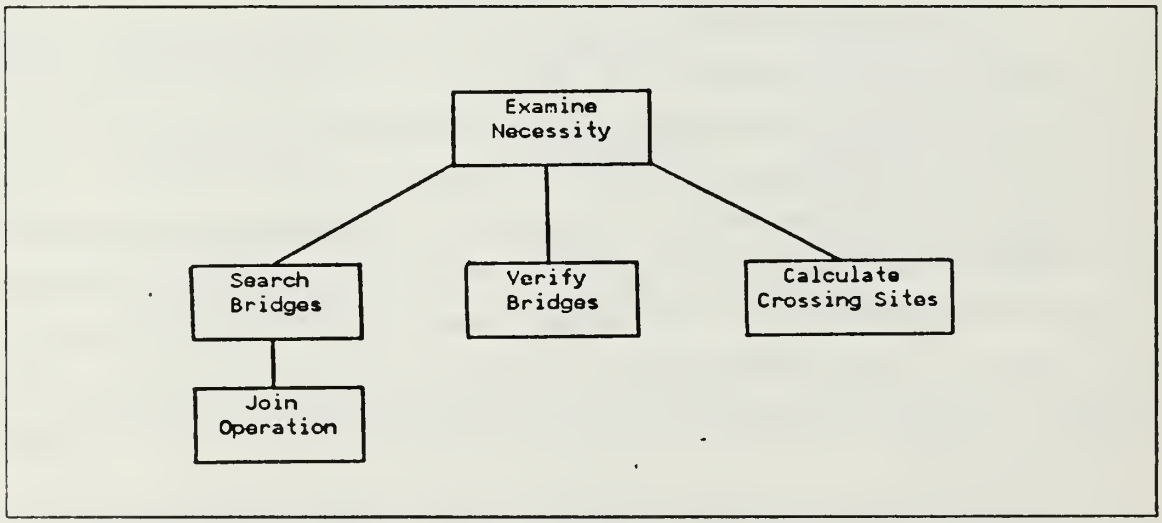

Figure 4.4 Decomposed Necessity Module.

b. Module 2 : Examine Possibility

When the River Crossing Operation Subsystem gets control back from the Module 1, Examine Necessity, it invokes the next module, Examine Possibility, to see if river crossing operation is necessary from the result of Module 1. This module estimates the possibility of the river crossing operation, using the command's engineer units and their equipments. In order to estimate the possibility, several functions such as Select Crossing Sites, Unit/EQPs, and Calculate Required Equipments are needed. First, the module has to select the crossing sites of the database. To select crossing sites, the module first surveys all the crossing sites in this area, then sort them by width of the river, and finally selects the crossing sites that have minimum widths and also with water velocity not over 2 mps(meter per second). The system has to consider the amount of equipments available for this operation. To do this, it examines the participating engineer units from the database file called ASSIGN. After examining these engineer units, the system surveys the available equipments of these unit from the database file called UNITEQP. Next, the module calculates the equipments requirement for these sites. Before calculating equipments, the module has to estimate the number of rafts per each raft site, which depends on the width of the river. Finally, the module estimates the possibility by comparing the required equipments to the available equipments. Therefore we can decomposed this module as in Figure 4.5.

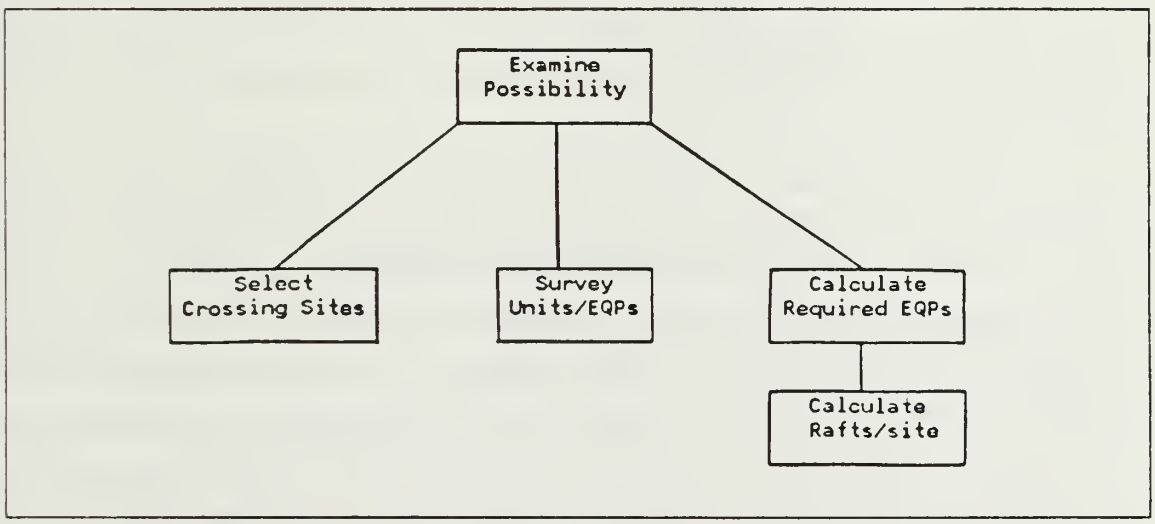

Figure 4.5 Decomposed Possibility Module.

c. Module 3 : Search Supporting Units

From the result of Module 2, the system searches available supporting units, if the river crossing operation is impossible without other engineer units' support and their equipments. The system invokes this module to find the available engineer units and equipments. First, the module determines the name of high-level command of the river crossing command from the database file called ASSIGN. Next the module determines which units are assigned to the high-level command, and after that, it determines which engineer units are assigned to these units. Finally it determines the assigned engineer units and their equipments.

d. Module 4 : Generate Crossing Capacity

The system invokes this module to generate the output to show how many vehicles can cross the river every hour and when the command and its vehicles and equipments finish crossing(H-hour). In order to do this, the module calculates how many trips are possible per hour at each crossing sites by raft and also calculates the assembly time for the rafts and floating bridges. Based on these calculations, the module generates the crossing capacity. Figure 4.6 shows the decomposed river crossing operation subsystem. The detailed algorithmes are given in Appendix A, which shows the Pseudo Programming of the proposed system.

2. **Subsystem 2 : Obstacle/Denial Operation**

We have already developed the logical model of this system in Chapter III. In this section, we convert the logical model of this system to physical model. Therefore we can develop a high-level hierarchy chart for this system based on the data flow diagram and system flowchart as in Figure 4.7.

This subsystem has four functions, Get Data, Examine Necessity, Select Targets, and Extend Segment of operational boundary. Further, this subsystem begins with initial data about road number, segment of road and required amount of delay time to prepare the next operation. This kind of data will be given to the subsystem through the Get Data module. After receiving the data, the subsystem invokes the Examine Necessity module to estimate the necessity of operation under the current situation. In order to estimate it, this module needs several functions such as Search Targets function that finds the designated targets in the operational boundary, Survey Target function which surveys the unemplaced or undenied targets among the targets that are already found by the previous function, and Calculate Delay Time function. After getting the initial data, the obstacle/denial operation invokes the Examine

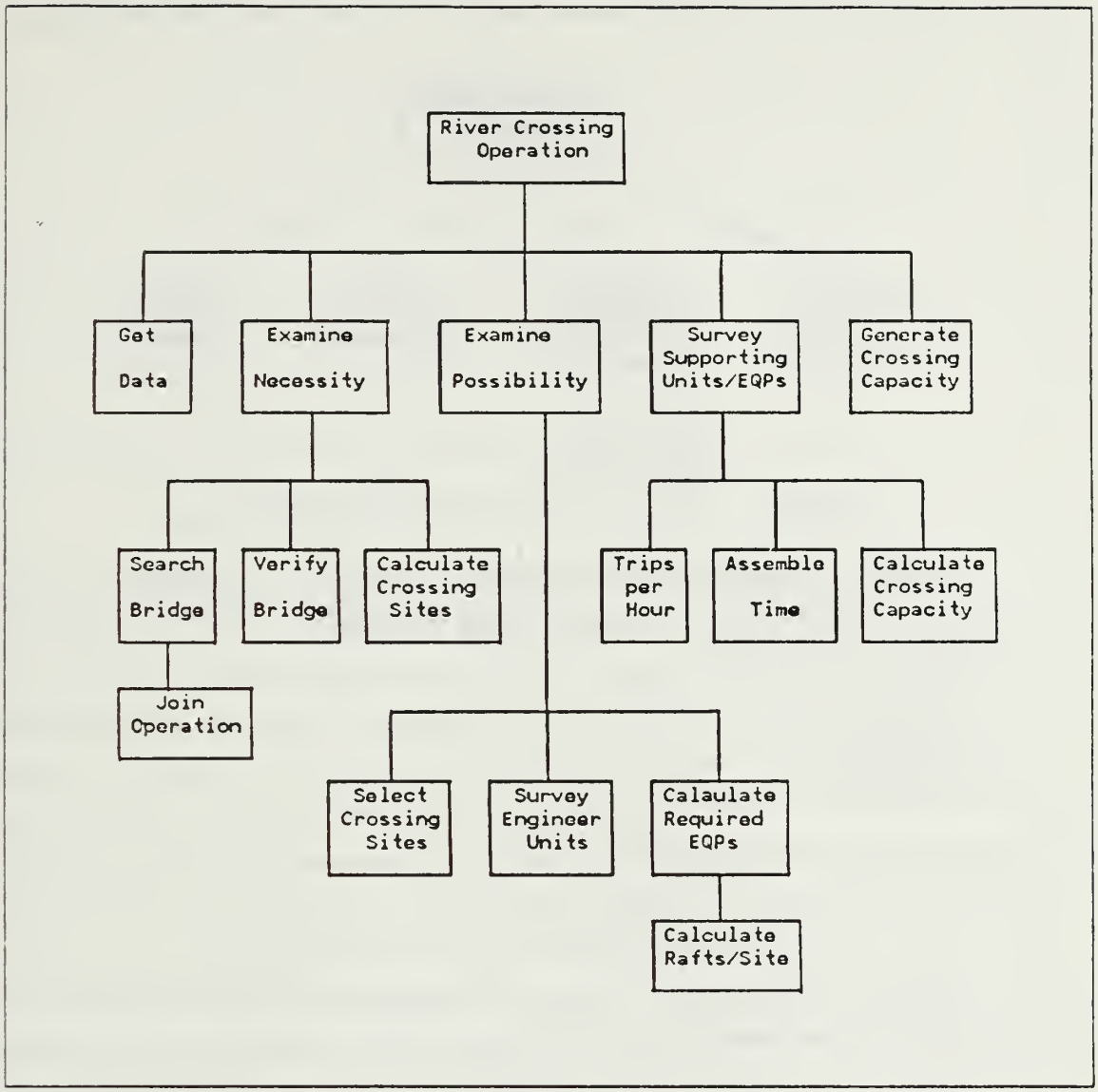

Figure 4.6 Decomposed River Crossing Operation Subsystem.

Necessity module to estimate the necessity of the obstacle/denial operation in the current situation. Because if the delay time of the already emplaced obstacles is enough to satisfy the commander's required delay time, then we don't need to conduct this operation in this area. Therefore this module needs several functions such as Search Target that searches the targets in the operational segment, Survey Target that examines the state of the targets, and Calculate Delay Time that calculates the time and generates the output of this module.

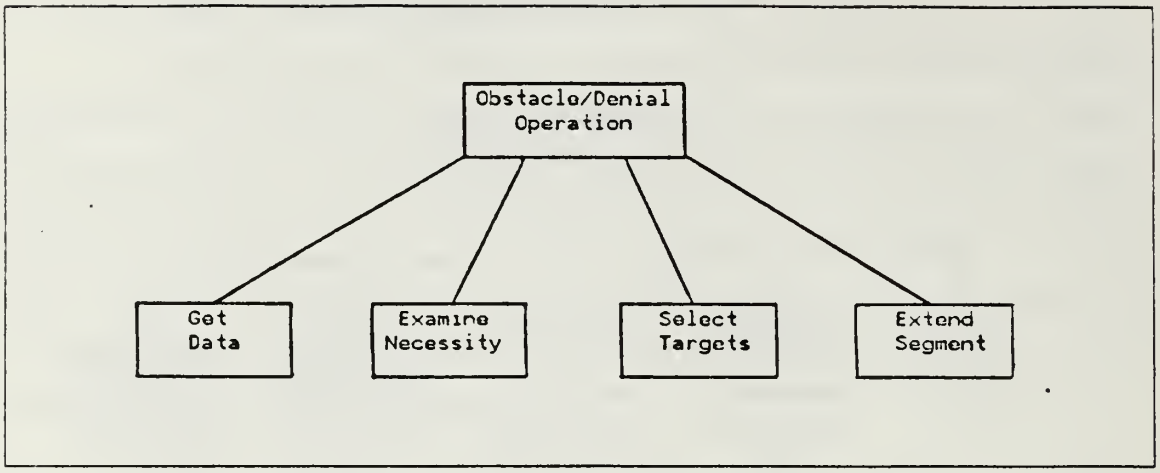

Figure 4.7 High Level Hierarchy Chart of Subsystem 2.

When the calculated delay time is not enough to satisfy the commander's requirement, the system call to the next module Select Target becomes necessary. This module selects the additional targets that will be emplaced as obstacles. At this time, the commander's selecting criteria to select the target as already discussed in the system analysis phase is required when the available delay time in this segment is not enough for the commander's requirement, the system finally calls the Extend Segment module in order to extend the operational boundary for this operation. This module is also functionally decomposed into several functions such as Determine Road Direction and Search Target. The Determine Road Direction Function examines the direction of road to extend the road segment. Although the coordinates are two dimensional value, X-axis and Y-axis, we need only one dimensional value to extend the road segment and search target. But sometimes we need both of X and Y value. From the result of these modules, the system can be functionally decomposed as in Figure 4.8. Appendix A shows a detailed algorithm of this system.

E. SYSTEM IMPLEMENTATION

The structured system analysis and design for the proposed system have been discussed in the previous sections. Specifications for the programs have been written. The time has come to discuss the implementation of the proposed system, as the final step of the structured system analysis and design methodology.

Implementation is simply the process of carrying out the specified design. It is concerned with coding, documenting, and debugging the programs. In addition, during

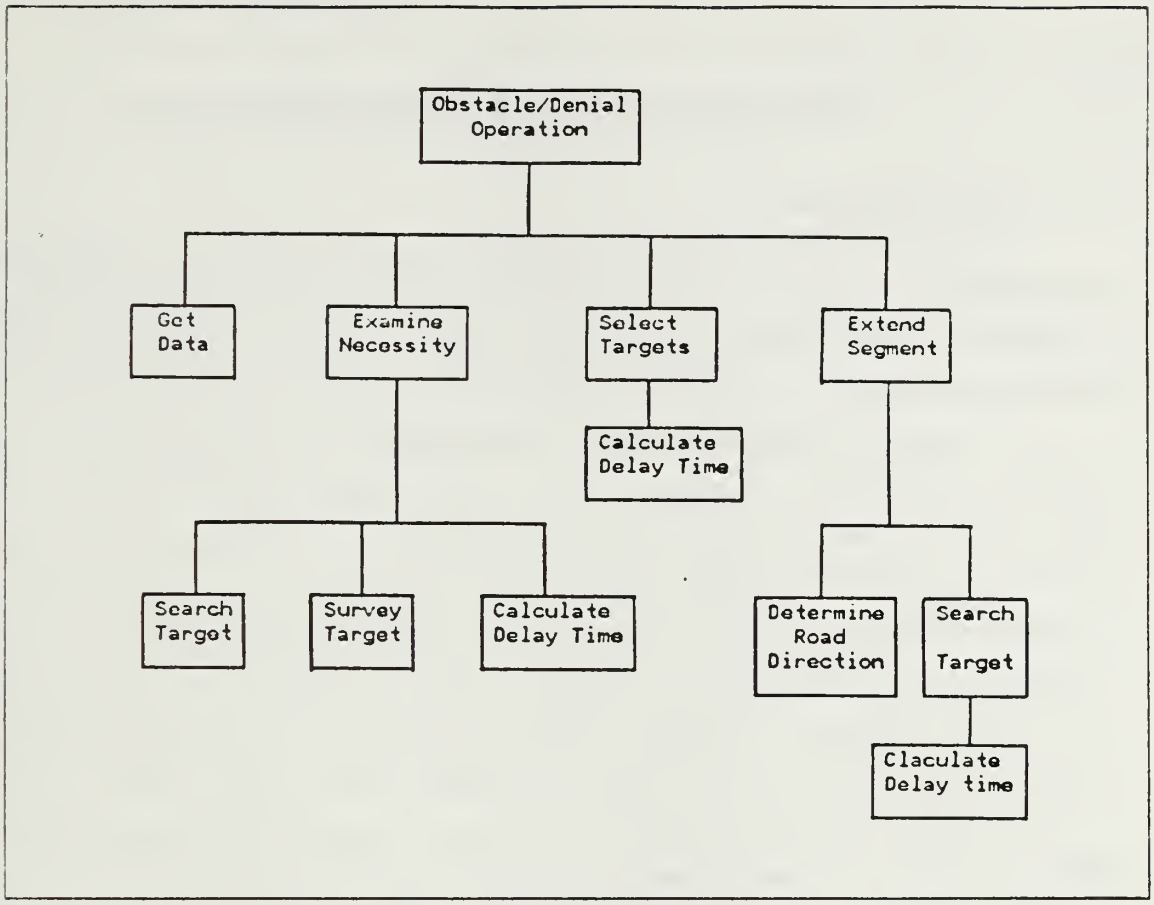

Figure 4.8 Decomposed Obstacle/Denial Subsystem.

implementation operating procedures are written. The need for training is an often overlooked aspect of system implementation. Further, system test is conducted in this step. [Ref. 7: p. 100]

The object of structured programming is to generate programs that are easy to understand, and hence easy to debug and maintain. The basic principle of structured programming is akin to the military strategy of divide and conquer. A program is broken into small, independent single functional modules which are clear and easy to follow. These modules are themselves composed of even smaller blocks, such as the sequence, decision, and repetition structures.

The river crossing operation subsystem of the proposed system has been implemented in a microcomputer using dBASE III Plus, as shown in Appendix B. Each module of this system is coded by top-down implementation technique using the hierarchical chart that has been developed in the system design phase.

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The goals of this thesis were to maintain and analyze information related to the Army Engineer Operation, and to provide information to the commander and his staffs in order to support their decision making process during the planning phase of an engineer operation.

To develop this system, the author first described the operational concepts of the engineer system under the combined arms team as the background of this system. After that, the author analyzed the current system of the Korean Army Engineer Operation to define the problems and defined the operational environments in order to determine the functions of this system to solve the problems. During the analysis and design process of this system, the structured system analysis and design methodology and artificial intelligence technique were used. A more general rule-based system architecture has not been used because of the restricted time to do the thesis, the desire to see and learn what appropriate information and system may be needed, and the belief that an operational system necessarily requires a complete re-do from this learning experience. Finally, the system was implemented as a prototype with the system dBASE III Plus.

The Engineer Operation Support System(EOSS) determines the necessity and possibility of the engineer operations, river crossing and obstacle/denial operations, during the planning phase of an operation. The commander and staff can save their decision making time using the result of this system as an output. In addition, they gain reliability of data and information related with engineer operation by using this system. To accomplish these objects, relevant data and information must be maintained accurately in the database file.

Through the study of this system, the author achieved the primary goals of this study by combining the knowledge of army operations, specifically army engineer operations, structured system analysis and design methodology, and programming techniques. This system is the most useful for the operations in the division and corps commands, but it can be used for the Field Army Command or Army Headquarters if the performance of the system can be improved and the amount of data to be stored is increased.

B. RECOMMENDATIONS

In this thesis, the author implemented the river crossing operation with dBASE III Plus as an example. But dBASE III Plus programs are slower than complied programs using C or Pascal. To improve the performance of this system, the system has to be rewritten in C or Pascal or some other more powerful logical programming languages like Prolog, and also the obstacle/denial operation part has to be implemented in the future. Furthermore, tactical situations such as the situations of the friendly and the enemy forces, fire support plan, movement plan, etc., have to be included in the future study of this system; here in this system only the engineer environment of the combined arms team has been considered.

APPENDIX A
ALGORITHMS

1. River Crossing Operation Subsystem

a. Get Data Module

```
Input Data
    MUNIT  : River crossing command
    MRNAME : Name of river
    LEFT   : Coordinate of leftmost of the crossing site
    MIDDLE : Coordinate of the center of the crossing site
    RIGHT  : Coordinate of rightmost of the crossing site
```

b. Examine Necessity Module

```
*** Search available bridge in area A and B ***
AREA = 'A', LOC1 = LEFT, LOC2 = MIDDLE
join with RIV_OBJ and BRIDGES to TEMP
      for RIV_OBJ.OTYPE = 'BRGE', RIV_OBJ.RNAME = MRNAME
          RIV_OBJ.LOC = BRIDGES.LOC
while .T. do
    select  LOC, BNAME, CLASS
    from    TEMP
    where   LOC1 <= TEMP.LOC <= LOC2, TEMP.STATE = 'AVAIL'
    insert to TEMP1
    if AREA = 'B' then exit loop
    else AREA = 'B', LOC1 = MIDDLE, LOC2 = RIGHT
    endif
end while
*** Calculate Crossing Site ***
BRG  = 1  :  Bridge site
RAFT = 3  :  Raft site
AREA = 'A'
while not eof(TEMP)
    if TEMP.CLASS >= 45 then BRG = BRG - 1
    else if TEMP.CLASS >= 20 then RAFT = RAFT - 2
          else RAFT = RAFT - 1
          endif
    endif
end while
if BRG <= 0 then BRG = 0
endif
if RAFT <= 0 then RAFT = 0
endif
*** Estimate Necessity ***
if BRG > 0 or RAFT > 0 then
    "River Crossing Operation is Necessary"
else "River Crossing Operation is not necessary"
endif
```

c. Select Crossing Sites

```
*** Selecting Crossing Sites***
AREA = 'A', LOC1 = LEFT, LOC2 = MIDDLE
join with RIV_OBJ and CROSSITE TO TEMP
      for RIV_OBJ.OTYPE = 'SITE', RIV_OBJ.RNAME = MRNAME
          RIV_OBJ.LOC = CROSSITE.LOC
while .T. do
    while not eof(TEMP)
        select LOC, WIDTH, VELO, OBST
        from   TEMP
        where  LOC1 <= TEMP.LOC <= LOC2
        insert to TEMP1
    end while
```

```
        sort TEMP1 with TEMP1.WIDTH(ascending order)
        while not eof(TEMP1) or ( BRG <> 0and RAFT <> 0 )
            select LOC, WIDTH
            from   TEMP1
            where  minimum WIDTH
                     VELO <= 2.0, OBST = 'none'
            insert to SELECTED
            BRG  = BRG  - 1, RAFT = RAFT - 1
        end while
        if AREA = 'B' then exit loop
        else AREA = 'B', LOC1 = MIDDLE, LOC2 = RIGHT
        endif
    end while
   *** Survey Organized Engineer Units and Equipments ***
    join with ASSIGN AND UNITEQP to TEMP
    select UNIT, AFB, LTR, M4T6
    from   TEMP
    where  TEMP.UNIT = MUNIT
    insert to TEMP1
    while not eof(TEMP)
        MAFB  = MAFB  + AFB
        MLTR  = MLTR  + LTR
        MM4T6 = MM4T6+ M4T6
    end while
   *** Calculate Required Equipment ***
    RAFB, RLTR, RM4T6  :  Required amount of equipments
    while not eof(SELECTED)
        if MEANS = "BRGE" then
            RM4T6 = RM4T6 + SELECTED.WIDTH / 43.2
        else if MEANS = "H.RAFT" then
                  case  WIDTH <= 100
                          RM4T6 = RM4T6 + 0.5
                  case  100 < WIDTH <= 200
                          RM4T6 = RM4T6 + 1.0
                  case  200 < WIDTH <= 300
                          RM4T6 = RM4T6 + 1.5
                  case  300 < WIDTH
                          RM4T6 = RM4T6 + 2.0
             else      (* MEANS = L.RAFT *)
                  case  WIDTH <= 100
                          RLTR = RLTR + 1
                  case  100 < WIDTH <= 200
                          RLTR = RLTR + 2
                  case  200 < WIDTH <= 300
                          RLTR = RLTR + 3
                  case  300 < WIDTH
                          RLTR = RLTR + 4
             endif
          endif
     end while
    *** Estimate Possibility ***
     if MAFB < RAFB or MLTR < RLTR or MM4T6 < RM4T6 then
        "Need support from other engineer unit"
     else "River Crossing Operation is possible"
     endif
```

d. Survey Supporting Unit Module

```
   *** search MUNIT from ASSIGN ***
    if found then HIGHER = ASSIGN.BELONG
        select  UNIT
        from    ASSIGN
        where   ASSIGN.BELONG = HIGHER
        insert to TEMP
        while not eof(TEMP)
            select  UNIT
            from    ASSIGN
            where   TWMP.UNIT = ASSIGN.BELONG
            insert to SUPPORT
```

```
    end while
    select  UNIT, AFB, LTR, M4T6
    from    UNITEQP
    where   SUPPORT.UNIT = UNITEQP.UNIT
else "There does not exist MUNIT"
endif
```

e. Generate Crossing Capacity Module

```
HV.HRAFT : Number of heavy vehicle that can be crossed by h.raft
HV.BRGE  : Number of heavy vehicle that can be crossed by bridge
LV.LTR   : Number of light vehicle that can be crossed by LTR
LV.HRAFT : Number of light vehicle that can be crossed by h.raft
LV.BRGE  : Number of light vehicle that can be crossed by bridge
H.TOTAL  : Total number of heavy vehicle that are already crossed
L.TOTAL  : Total number of light vehicle that are already crossed
AT.HRAFT : Heavy raft assemble time
AT.BRGE  : Bridge assemble time
AT.LTR   : LTR assemble time
L.HOUR   : Hours to cross light vehicle
HOUR     : Hours to cross heavy vehicle

HOUR = 0,       L.HOUR = 0,    LV.LTR = 0,
HV.HRAFT = 0,   HV.BRGE = 0,   H.TOTAL = 0
LV.HRAFT = 0,   LV.BRGE = 0,   L.TOTAL = 0

while HOUR <= AT.HRAFT or HOUR < AT.BRGE
    if HOUR > AT.HRAFT then
        HV.HRAFT = HV.HRAFT + (TRIPS * NUM.HRAFT )
        H.TOTAL = H.TOTAL + HV.HRAFT
        HOUR = HOUR + 1
    endif
end while

while H.TOTAL < H.VEH do
    HV.HRAFT = HV.HRAFT + ( TRIPS * NUM.HRAFT )
    HV.BRGE  = HV.BRGE  + ( 200 * NUM.BRGE )
    H.TOTAL  = H.TOTAL  + HV.HRAFT + HV.BRGE
    HOUR = HOUR + 1
end while

while AT.LTR >= L.HOUR do
    L.TOTAL = LV.LTR
    L.HOUR = L.HOUR + 1
end while

while L.TOTAL < L.VEH do
    LV.LTR = LV.LTR + ( TRIPS * NUM.LTR )
    if HOUR <= L.HOUR then
        LV.HRAFT = LV.HRAFT + ( TRIPS * NUM.HRAFT * 2 )
        LV.BRGE  = LV.BRGE  + ( 200 * NUM.BRGE )
    endif
    L.TOTAL = L.TOTAL + LV.LTR + LV.HRAFT + LV.BRGE
    L.HOUR = L.HOUR + 1
end while

if L.HOUR >= Hour then FINISHING TIME = L.HOUR
else FINISHING TIME = HOUR
```

2. OBSTACLE/DENIAL OPERATION SUBSYSTEM

a. Get Data Module

```
Input Data
    MRNUM  : Road number
    FROM   : Coordinate of starting point of road segment
    ENDED  : Coordinate of ending point of road segment
    RDTIME : Required delay time
```

b. Examine Necessity Module

```
*** Search planned targets in this segment of road ***
 join with RD_OBJ and TARGETS to TEMP
       for MRNUM = RD_OBJ.RNUM, RD_OBJ.OTYPE = 'OBST'
           RD_OBJ.LOC = TARGETS.LOC
 select TNUM, LOC, TTYPE, DTIME, STATE
 from   TEMP
 where  LOC1 <= TEMP.LOC <= LOC2
 insert to TEMP1
*** Calculate Delay Time ***
 SUM = 0
 while not eof(TEMP1)
    if TEMP1.STATE = 'done' then
       SUM = SUM + TEMP1.DTIME endif
 end while
*** Estimate Necessity ***
 if SUM >= RTIME then "Don't need Additional Operation"
 else ADDTIME = RTIME - SUM, "Need Additional Operation"
```

c. Select Targets Module

```
*** Input Option ***
 Target Type : Target that you want to be denied
 Selecting Criteria : MAX. and MIN. number of targets
*** Select Proper Target ***
 sort TEMP1 with Delay Time(ascending order)
 if TYPE = 'ALL' then
    if CRITERIA = MAX then go to the end of TEMP1
       while SUM < ADDTIME or not bof(TEMP)
          select TNUM, LOC, TYPE, DTIME
          from   TEMP1
          where STATE = 'plan'
          SUM = SUM + DTIME
       end while
    else (* CRITERIA = MIN *)
       while SUM < ADDTIME or not eof(TEMP1)
          select TNUM, LOC, TYPE, DTIME
          from   TEMP1
          where  STATE = 'plan'
          SUM = SUM + DTIME
       end while
    endif
 else
    while MORE = true
       if CRITERIA = MAX then goto bottom
          while SUM < ADDTIME or not bof
             select TNUM, LOC, TYPE, DTIME
             from   TEMP1
             where  STATE = 'plan', TYPE = Input target type
             SUM = SUM + DTIME
          end while
       else (* CRITERIA = MIN *)
          while SUM < ADDTIME or not eof(TEMP1)
             select TNUM, LOC, TYPE, DTIME
             from   TEMP1
             where  STATE = 'plan', TYPE = Input target type
             SUM = SUM + DTIME
          end while
       endif
       if SUM < ADDTIME then "Need other type of targets"
```

```
                MORE = true
            else MORE = folse
            endif
        end while
    endif
    if SUM < ADDTIME then "Impossible within this segment"
        call EXTEND SEGMENT
    else DISPLAY OUTPUT
    endif
```

d. Extend Segment Module

```
*** Determine the Direction of Road ***
 AX = abs ( X.LOC1 - X.LOC2 )
 AY = abs ( Y.LOC1 - Y.LOC2 )
 if AX > AY then "Direction of road : East to West"
 else "Direction of road : South to North" endif
*** Search Targets ***
 if AX > AY then
     if X.LOC1 > X.LOC2 then
         while SUM < ADDTIME or bof(TEMP)
             select TNUM, LOC, TYPE, DTIME, STATE
             from   INDEXED TEMP
             where X.LOC < X.LOC2
             SUM = SUM + DTIME
         end while
     else
         while SUM < ADDTIME or eof(TEMP)
             select TNUM, LOC, TYPE, DTIME, STATE
             from   INDEXED TEMP
             where  X.LOC > X.LOC2
             SUM = SUM + DTIME
         end while
     endif
 else if Y.LOC1 > Y.LOC2 then
       while SUM < ADDTIME or bof(TEMP)
           select TNUM, LOC, TYPE, DTIME, STATE
           from   INDEXED TEMP
           where  Y.LOC < Y.LOC2
           SUM = SUM + DTIME
       end while
       else while SUM < ADDTIME or eof(TEMP)
                 select TNUM, LOC, TYPE, DTIME, STATE
                 from   INDEXED TEMP
                 where  Y.LOC > Y.LOC2
                 SUM = SUM + DTIME
             end while
       endif
 endif
```

APPENDIX B
PROGRAM LISTING OF THE RIVER CROSSING OPERATION

1. Main Program

```
*********************************************************************
****                  E N G I N E E R . P R G                    ****
*********************************************************************
* Module name : ENGINEER.PRG                                        *
* Author      : Jang, Chang Ki                                      *
* Purpose     : This is a Engineer Operation System to support      *
*               the decision making during the Army Engineer Op-    *
*               eration. This module calls the submodules of this   *
*               system to execute the whole system.                 *
* Called by   : System start up (dBASE III Plus)                    *
* Calls       : MENUSCR, RIVER, DENIAL, FMODIFY                     *
* Variables                                                         *
*      Global : OPTION : Holds the user's input option              *
*               TODAY  : Date of today                              *
*      Local  : none                                                *
*********************************************************************
*------------------------------------------ Close all open files
clear all
*---------------------------------------- Set working environment
set talk off
set heading off
set safety off
set status off
*------------------------------------------ Define public variable
public TODAY, OPTION
store space(1) to OPTION
store data() to TODAY
*-------------------------------------------------------------------*
*---- This sets up the CRASH.TXT file which records all actions so
*---- that if the system crashes the data base can be recreated.
*---- This files is deleted if the system terminates normally.
*-------------------------------------------------------------------*
set alte to crash
set alte on
do while .T.
    *------------------- Clear the screen and display the main menu
    clear
    do MENUSCR
    set color to R+
    @ 2,15 say " A R M Y   E N G I N E E R   O P E R A T I O N "
    @ 6,33 say "M E N U"
    @19,33 say "INFORMATION"
    set color to G
    @ 9,22 say "R  :  RIVER CROSSING OPERATION"
    @11,22 say "D  :  DENIAL (OBSTACLE) OPERATION"
    @13,22 say "M  :  DATA FILE MODIFY"
    set color to R+
    @15,22 say "X  :  EXIT FROM RUNNING"
    set color to W
    @21,58 say time()
    set color to R+
    @23,15 "  Enter selection ( R, D, M, or X to Exit ) :   "
    set color to W
    @23,61 say "" get OPTION
    read
    store upper(OPTION) to OPTION
    do while .T.
        do case
```

```
            case OPTION = 'X'
                  clear all
                  quit
            case OPTION = 'R'
                  do RIVER
                  exit
            case OPTION = 'D'
                  do DENIAL
                  exit
            case OPTION = 'M'
                  do FMODIFY
                  exit
            *----------------------- Error condition of user's option
            otherwise
                  @20, 8 say space(18)
                  @20,60 say space(5)
                  @21, 5 say space(62)
                  set color to R+
                  @21,20 say "ERROR MESSAGE---ILLEGAL OPTION :"+OPTION
                  set color to W
                  ?? chr(7)
                  @23,61 say "" get OPTION
                  @23,62 say space(1)
                  read
                  store upper(OPTION) to OPTION
            endcase
      *------------------------------------------------ End of loop
      enddo
      *------------------------------------------- End of main loop
enddo
set alte off
clear all
close alte
erase CRASH.TXT
*=================    END  OF  MAIN PROGRAM    ==================*


*************************************************************************
****                    M E N U S C R . P R G                        ****
*************************************************************************
* Module name : MENUSCR.PRG                                             *
* Purpose     : This is a MENUSCR module of Engineer Operation          *
*               system to format the screen.                            *
* Called by   : ENGINEER                                                *
*************************************************************************
TODAY = date()
@  1, 9 to  3,69
@  4, 1 to 24,77 double
@  6, 3 to 17,75
@  5,30 to  7, 4 double
@  6,31 say space(15)
@ 19, 3 to 22,75
@ 18,30 to 20,46
@ 19,31 say space(15)
@  5, 2 say replicate(chr(176),28)
@  6, 2 say chr(176)
@  7, 2 say chr(176)
@  8, 2 say chr(176)
@  9, 2 say chr(176)
@ 10, 2 say chr(176)
@ 11, 2 say chr(176)
@ 12, 2 say chr(176)
@ 13, 2 say chr(176)
@ 14, 2 say chr(176)
@ 15, 2 say chr(176)
@ 16, 2 say chr(176)
@ 17, 2 say chr(176)
@ 18, 2 say replicate(chr(176),28)
```

```
@ 19, 2 say chr(176)
@ 20, 2 say chr(176)
@ 21, 2 say chr(176)
@ 22, 2 say chr(176)
@ 23, 2 say replicate(chr(176),75)
@ 22,76 say chr(176)
@ 21,76 say chr(176)
@ 20,76 say chr(176)
@ 19,76 say chr(176)
@ 18,47 say replicate(chr(176),30)
@ 17,76 say chr(176)
@ 16,76 say chr(176)
@ 15,76 say chr(176)
@ 14,76 say chr(176)
@ 13,76 say chr(176)
@ 12,76 say chr(176)
@ 11,76 say chr(176)
@ 10,76 say chr(176)
@  9,76 say chr(176)
@  8,76 say chr(176)
@  7,76 say chr(176)
@  6,76 say chr(176)
@  5,47 say replicate(chr(176),30)
@ 19,33 say "INFORMATION"
@ 20,14 say "DATE"
@ 20,60 say "TIME"
@ 21,12 say today
@ 21,58 say time()
return
*======================    END OF MENUSCR    ====================*
```

2. River Crossing Operation Subsystem

```
*************************************************************************
****                        R I V E R . P R G                        ****
*************************************************************************
* Module name : RIVER.PRG                                               *
* Purpose     : This is a River Crossing Operation Support Sys-         *
*               tem, this system controls the overall subsystem         *
*               of this system.                                         *
* Called by   : ENGINEER                                                *
* Calls       : MENUSCR, GETDATA, NECESSRC, POSSIBLE, SUPPORT           *
*               GENERATE                                                *
* Variables                                                             *
*      Global : MUNIT, MRNAME, LEFT, RIGHT, MIDDLE, A_BRG,              *
*               B_BRG, A_RAFT, B_RAFT, MAFB, MLTR, MM4T6,               *
*               RAFB, RLTR, RM4T6                                       *
*               TODAY  : Date of today                                  *
*      Local  : none                                                    *
*************************************************************************

*-------------------------------------------- Clear all open file
clear all
set talk off
do while .T.
    *---------------------------------- Define the global variables
    *----- MUNIT           : Name of the river crossing command
    *----- MRNAME          : Name of the river to cross
    *----- LEFT            : Coordinate of left boundary of area
    *----- RIGHT           : Coordinate of right boundary of area
    *----- MIDDLE          : Coordinate of center of crossing area
    *----- A_BRG, B_BRG    : Number of bridge sites in area A and B
    *----- A_RAFT, B_RAFT  : Number of raft sites in area A and B
    *----- MAFB,MLTR,MM4T6 : Available amount of equipments
    *----- RAFB,RLTR,RM4T6 : Required amount of equipments
    *----- DECISION        : User's decision
    public MUNIT, MRNAME, LEFT, RIGHT, MIDDLE, DECISION, OPTION,;
```

```
            A_BRG, B_BRG, A_RAFT, B_RAFT, MAFB, MLTR, MM4T6, RAFB,;
            RLTR, RM4T6
    store space(6)  to MUNIT
    store space(10) to MRNAME
    store space(8)  to LEFT
    store space(8)  to RIGHT
    store space(1)  to OPTION, DECISION
    store 0 to A_BRG, B_BRG, A_RAFT, B_RAFT
    store 0 to MAFB, MLTR, MM4T6, RAFB, RLTR, RM4T6
    do while .T.
        *------------ Call GETDATA to define operational environment
        do GETDATA
        if OPTION = 'X'
            clear all
            return
            *------------------------ Return to main menu of ENGINEER
        endif
        *---------------------- Call NECESSRS to estimate necessity
        do NECESSRC
        if OPTION = 'X'
            clear all
            exit
            *------------------------------------- Back to the RIVER
        endif
        *--------------------- Call POSSIBLE to estimate possibility
        do POSSIBLE
        if OPTION = 'Y'
            do SUPPORT
        endif
        if OPTION = 'X'
            clear all
            exit
            *------------------------------------- Back to the RIVER
        endif
        *------------------ Call GENERATE to estimate finishing time
        if OPTION = 'X'
            clear all
            exit
            *------------------------------------- Back to the RIVER
        endif
    *------------------------------------------------ end of loop
    enddo
return
*====================    END  OF  RIVER.PRG     ==================*
```

3. Get Data Module

```
*********************************************************************
****                    G E T D A T A . P R G                    ****
*********************************************************************
* Module name : GETDATA.PRG                                         *
* Purpose     : This is a GETDATA module to define operational      *
*               environments such as name of river crossing co-     *
*               mmand, name of river, and river crossing area.      *
* Called by   : RIVER                                               *
*********************************************************************

set talk off
clear
*---------------------------- Call MENUSCR to formating the screen
do MENUSCR
do while .T.
    erase TEMP1
    set color to R+
    @  2,14 say "R I V E R   C R O S S I N G   O P E R A T I O N "
    @  6,34 say "INITIALIZE"
    @ 19,33 say "INFORMATION"
```

```
    set color to G
    @  9,21 say "RIVER CROSSING COMMAND  : "
    @ 11,21 say "NAME OF RIVER TO CROSS  : "
    @ 13,21 say "CO-ORDINATES OF RIVER CROSSING AREA"
    @ 15,13 say "LEFT"
    @ 15,31 say "MIDDLE"
    @ 15,50 say "RIGHT"
    set color to W
    @ 21,58 say time()
    set color to R+
    @ 23,15 say "  Press any key to RUN or Press 'X' to EXIT :  "
    @ 23,62 say "" get OPTION
    read
    set color to W
    store upper(OPTION) to OPTION
    if OPTION = 'X'
       @ 23,62 say ' '
       return
       *---------------------------- Back to main menu of ENGINEER
    endif
    do while .T.
       @  9,48 say "" get MUNIT
       read
       store upper(MUNIT) to MUNIT
       @ 11,48 say "" get MRNAME
       read
       store upper(MRNAME) to MRNAME
       @ 15,20 say "" get LEFT
       read
       store upper(LEFT) to LEFT
       @ 15,39 say "" get MIDDLE
       read
       store upper(MIDDLE) to MIDDLE
       @ 15,57 say "" get RIGHT
       read
       store upper(RIGHT) to RIGHT
       set color to R+
       @ 16,18 say "Press 'C' change data or Press any key  : "
       set color to W
       @ 16,62 say "" get OPTION
       READ
       store upper(OPTION) to OPTION
       if OPTION <> 'C'
          exit
       endif
    enddo
    @ 23,62 say "" get OPTION
    read
    store upper(OPTION) to OPTION
    if OPTION = 'X'
       return
       *---------------------------- Back to main menu of ENGINEER
    else
       exit
    endif
enddo
return
*==================     END  OF GETDATA.PRG     =================*
```

4. Necessity Module

```
*****************************************************************
****                  N E C E S S R C . P R G                 ****
*****************************************************************
* Module name : NECESSRC.PRG                                    *
* Purpose     : This is a NECESSRC module to estimate the nece- *
*               ssity of river crossing operation.              *
* Called by   : RIVER                                           *
* Calls       : EXISTBRG, CALCSITE                              *
* Variables                                                     *
*      Local  : AREA, LOC1, LOC2, ANEED, BNEED, MX, MLINE,      *
*               BRGESITE, RAFTSITE                              *
*****************************************************************
set talk off
clear
*------------------------------------ Define the local variables
*-------- AREA          : Area of river crossing operation
*-------- LOC1, LOC2    : Coordinates of river crossing area
*-------- ANEED, BNEED  : Necessity of operation for area A and B
*-------- BRGESITE      : Required number of bridge sites
*-------- RAFTSITE      : Required number of raft sites
*-------- MLINE         : Line control variable for output
*-------- MX            : Pointer variable for data base files
store space(1) to AREA
store space(8) to LOC1, LOC2
store 1 to A_BRGE, B_BRGE,MX
store 3 to A_RAFT, B_RAFT
store .T. to ANEED, BNEED
store 0 to BRGESITE, RAFTSITE
store 'A' to AREA
store 7 to MLINE
store LEFT to LOC1
store MIDDLE to LOC2
*------------------------------------ Define the working area
select A
use RIV_OBJ index RIV_OBJ
select B
use BRIDGES index BRIDGES
*--------------------- TEMP contains the all bridges in this river
join with RIV_OBJ to TEMP for B-> LOC = A-> LOC .and.;
      A-> RNAME = MRNAME fields LOC, BNAME, CLASS, STATE
copy structure to TEMP1
do while .T.
    *------------------------- Call existing bridge module EXISTBRG
    do EXISTBRG WITH AREA, ANEED, BNEED, MLINE, MX
    if AREA = 'B'
       exit
    endif
    AREA = 'B'
    LOC1 = MIDDLE
    LOC2 = RIGHT
enddo
BRGESITE = A_BRG  + B_BRG
RAFTSITE = A_RAFT + B_RAFT
if BRGESITE = 2 .and. RAFTSITE = 6
    @  6,12 to 8,62
    set color to R+
    @  7,17 "THERE IS NO AVAILABLE BRIDGE IN THIS AREA"
    set color to W
endif
if .not. (ANEED) .and. .not. (BNEED)
    @ 17,10 to 20,60 double
    set color to R+
    @ 18,15 say "RIVER CROSSING OPERATION IS NOT NECESSARY"
    set color to W
else
    @ 13,17 to 15,57
```

```
    set color to R+
    @ 14,20 say "REQUIRED  NUMBER  OF  CROSSING  SITES"
    set color to W
    @ 16,16 say "SITE"
    @ 16,31 say "A"
    @ 16,42 say "B"
    @ 16,53 say "TOTAL"
    @ 17,16 say replicate("=", 4)
    @ 17,30 say replicate("=", 3)
    @ 17,41 say replicate("=", 3)
    @ 17,53 say replicate("=", 5)
    @ 19,16 say "RAFT"
    @ 19,31 say ltrim(str(A_RAFT))
    @ 19,42 say ltrim(str(B_RAFT))
    @ 19,55 say ltrim(str(RAFTSITE))
    @ 21,16 say "BRIDGE"
    @ 21,31 say ltrim(str(A_BRG))
    @ 21,42 say ltrim(str(B_BRG))
    @ 21,55 say ltrim(str(BRGESITE))
endif
set color to R+
@ 23,14 say "Press any key to continue or Press "X" to EXIT : "
@ 23,64 say "" get OPTION
read
set color to W
store upper(OPTION) to OPTION
return
*=================     END  OF  NECESSRC.PRG     ================*


*********************************************************************
****                    E X I S T B R G . P R G                  ****
*********************************************************************
* Module name : EXISTBRG.PRG                                        *
* Purpose     : This is a sub-module of NECESSRC to examine the     *
*               existing available bridges in this area.            *
* Called by   : NECESSRC                                            *
* Variables                                                         *
*      Local  : O, P, Q, R, S, T, U, V, W, X, Y, Z, AX, AY          *
*********************************************************************

parameters AREA, LOC1, LOC2
*---------------- Define the local variables to represent location
store upper(substr(LOC1,1,2)) to O
Q = val(substr(LOC1,3,3))
S = val(substr(LOC2,3,3))
T = val(substr(LOC1,6,3))
V = val(substr(LOC2,6,3))
W = val(substr(LEFT,3,3))
X = val(substr(RIGHT,3,3))
Y = val(substr(LEFT,6,3))
Z = val(substr(RIGHT,6,3))
AX = abs(W - X)
AY = abs(Y - Z)
*---------------------------------------- Define the working area
select A
use TEMP
select B
use TEMP1
select TEMP
*------------------------ Searching the available existing bridgs
*------------------------ TEMP1 contains the available bridges
do while .T.
    store upper(substr(LOC,1,2)) to P
    R = val(substr(LOC,3,3))
    U = val(substr(LOC,6,3))
    do case
        case AX < AY
            if P = O .and. STATE = 'AVAIL' .and.;
```

```
            ((T <= U .and. U <= V) .or. (V <= U .and. U <= T))
            select TEMP1
            append blank
            replace LOC with TEMP->LOC, BNAME with TEMP->BNAME,;
                CLASS with TWMP->CLASS, STATE with TEMP->STATE
        endif
    case AX = AY
        if P = 0 .and. STATE = 'AVAIL' .and.;
            ((T <= U .and. U <= V) .or. (V <= U .and. U <= T))
            select TEMP1
            append blank
            replace LOC with TEMP->LOC, BNAME with TEMP->BNAME,;
                CLASS with TWMP->CLASS, STATE with TEMP->STATE
        else
            if P = 0 .and. STATE = 'AVAIL' .and.;
                ((Q <= R .and. R <= S) .or. (S <= R .and. R <= Q))
                select TEMP1
                append blank
                replace LOC with TEMP->LOC, BNAME with TEMP->BNAME,;
                    CLASS with TWMP->CLASS, STATE with TEMP->STATE
            endif
        endif
    case AX > AY
        if P = 0 .and. STATE = 'AVAIL' .and.;
            ((Q <= R .and. R <= S) .or. (S <= R .and. R <= Q))
            select TEMP1
            append blank
            replace LOC with TEMP->LOC, BNAME with TEMP->BNAME,;
                CLASS with TWMP->CLASS, STATE with TEMP->STATE
        endif
endcase
select TEMP
skip
enddo
close all
return
*=================     END  OF  EXISTBRG.PRG     ================*



*********************************************************************
****                    C A L C S I T E . P R G                  ****
*********************************************************************
* Module name : CALCSITE.PRG                                        *
* Purpose     : This is a sub-module of NECESSRC to calculate the*
*               required number of crossing sites.                  *
* Called by   : NECESSRC                                            *
*********************************************************************

parameters AREA, ANEED, BNEED, MLINE, MX
set talk off
use TEMP1
if MLINE = 7 .and. reccount() > 0
    @  1,24 to 3,50
    set color to R+
    @  2,28 say "AVAILABLE   BIDGES"
    set color to W
    @  4,16 say "AREA"
    @  4,25 say "LOCATION"
    @  4,38 say "BRIDGE NAME"
    @  4,54 say "CLASS"
    @  5,16 say replicate("=", 4)
    @  5,25 say replicate("=", 8)
    @  5,38 say replicate("=",11)
    @  5,54 say replicate("=", 5)
endif
use TEMP1
do while MX < reccount()
    goto MX
```

```
    @ MLINE,18 say AREA
    @ MLINE,25 say LOCA
    @ MLINE,40 say BNAME
    @ MLINE,56 say CLASS
    MLINE = MLINE + 1
    if AREA = "A"
        if A_BRG = 0
            if A_RAFT = 0
                ANEED = .F.
            else
                if CLASS >= 20
                    A_RAFT = A_RAFT - 2
                else
                    A_RAFT = A_RAFT - 1
                endif
            endif
        else
            if CLASS >= 45
                A_BRG = 0
            else
                if CLASS >= 20
                    A_RAFT = A_RAFT - 2
                else
                    A_RAFT = A_RAFT - 1
                endif
            endif
        endif
    else
        if B_BRG = 0
            if B_RAFT = 0
                BNEED = .F.
            else
                if CLASS >= 20
                    B_RAFT = B_RAFT - 2
                else
                    B_RAFT = B_RAFT - 1
                endif
            endif
        else
            if CLASS >= 45
                B_BRG = 0
            else
                if CLASS >= 20
                    B_RAFT = B_RAFT - 2
                else
                    B_RAFT = B_RAFT - 1
                endif
            endif
        endif
    endif
    MX = MX + 1
enddo
if A_BRG <= 0
    A_BRG = 0
endif
if B_BRG <= 0
    B_BRG = 0
endif
if A_RAFT <= 0
    A_RAFT = 0
endif
if B_RAFT <= 0
    B_RAFT = 0
endif
return
*==================     END  OF  CALCEPQS.PRG     ================*
```

5. Possibility Module

```
*****************************************************************
****                  P O S S I B L E . P R G                ****
*****************************************************************
* Module name : POSSIBLE.PRG                                    *
* Purpose     : This is a POSSIBLE module to estimate the poss- *
*               ibility of river crossing operation with organ- *
*               ized engineer units and their equipments.       *
* Called by   : RIVER                                           *
* Calls       : SELSITE, UNITEQPS, CALCEQPS                     *
* Variables                                                     *
*      Local  : MOREAFB, MORELTR, MOREM4T6                      *
*****************************************************************
set talk off
clear
*------------------------------------ Define the local variables
*---------------- MOREAFB  : Lack of AFB
*---------------- MORELTR  : Lack of LTR
*---------------- MOREM4T6 : Lack of M4T6
*---------------- NEED     : Boolean variable
store 0.0 to MOREAFB, MORELTR, MOREM4T6
store .F. to NEED
*---------------- Calls SELSITE to select the river crossing sites
do SELSITE
if OPTION = "X"
    return
    *------------- Return to RIVER.PRG
endif
*---------------- Calls UNITEQPS to examine the organized engineer
*---------------- units and their river crossing equipments.
do UNITEQPS
if OPTION = "X"
    return
    *------------- Return to RIVER.PRG
endif
*------------- Calls CALCEQPS to calculate the required equipments
do CALCEQPS
if OPTION = "X"
    return
    *------------- Return to RIVER.PRG
endif
if RAFB > MAFB
    NEED = .T.
    MOREAFB = RAFB - MAFB
endif
if RLTR > MLTR
    NEED = .T.
    MORELTR = RLTR - MLTR
endif
if RM4T6 > MM4T6
    NEED = .T.
    MOREM4T6 = RM4T6 - MM4T6
endif
@  2,23 to 4,54
@  3,27 say "STATUE   OF   EQUIPMENTS"
@  5,32 say "AFB"
@  5,44 say "LTR"
@  5,56 say "M4T6"
@  6,31 say replicate("=", 5)
@  6,43 say replicate("=", 5)
@  6,56 say replicate("=", 4)
@  8,14 say "AVAILABLE"
@  8,32 say str(MAFB,3,1)
@  8,44 say str(MLTR,3,1)
@  8,56 say str(MM4T6,3,1)
@ 10,14 say "REQUIRED"
@ 10,32 say str(RAFB,3,1)
```

```
@ 10,44 say str(RLTR,3,1)
@ 10,56 say str(RM4T6,3,1)
@ 14,16 say "LACK"
@ 14,32 say str(MOREAFB,3,1)
@ 14,44 say str(MORELTR,3,1)
@ 14,56 say str(MOREM4T6,3,1)
set color to R+
if NEED
    @ 18,17 say "NEED ADDITIONAL EQUIPMENT TO SUCCESS !!"
    @ 21,14 say "Do you want to search supporting unit? (Y/N) : "
    @ 21,63 say "" get OPTION
    read
    store upper(OPTION) to OPTION
    if OPTION = "Y"
        return
        *------------------------- Return to RIVER.PRG
    endif
else
    @ 17,16 say "WE HAVE ENOUGH EQUIPMENTS TO CONDUCT OPERATION"
endif
@ 23,14 say "Press any key to continue or Press "X" to EXIT : "
set color to R+
@ 23,64 "" get OPTION
read
store upper(OPTION) to OPTION
return
*=================     END  OF  POSSIBLE.PRG     ================*


*********************************************************************
****                  S E L S I T E . P R G                      ****
*********************************************************************
* Module name : SELSITE.PRG                                         *
* Purpose     : This is a sub-module of POSSIBLE to select the      *
*               river crossing sites in each area A and B.          *
* Called by   : POSSIBLE                                            *
* Calls       : INAREA, CHOOSING                                    *
*********************************************************************

set talk off
clear
use CROSSITE
copy structure to TEMP
copy structure to TEMP_A
copy structure to TEMP_B
copy structure to TEMP_C
use RIV_OBJ
*------------ TEMP1 contains the all crossing sites in this river
copy to TEMP1 for OTYPE = 'SITE' .and. RNAME = MRNAME
use TEMP1
index on LOC to TEMP1
*---------------------------------------- Define the working area
select 1
use TEMP
select 2
use TEMP1 index TEMP1
select 3
use CROSSITE index CORSSITE
set relation to LOC into TEMP1
*-------------------- Join with TEMP1 and CROSSITE to TEMP
*-------------------- TEMP contains the characteristics of all
*-------------------- river crossing sites in this river.
do while .not. eof()
    select temp1
    if .not. eof()
        select TEMP
        append blank
        replace LOC with TEMP1->LOC, width with CROSSITE->WIDTH,;
            DEPTH with CROSSITE->DEPTH, VELO with CROSSITE->VELO,;
            OBSTAC WITH CROSSITE->OBSTAC
```

```
    endif
    select CROSSITE
    skip
enddo
close all
*---------------- Calls INAREA to search the crossing sites
*---------------- in the river crossing area.
do INAREA
@  3,21 to 5,55
set color to R+
@  4,25 say "SELECTED   CROSSING   SITES"
set color to W
@  7,13 say "AREA"
@  7,22 say "LOCATION"
@  7,35 say "MEANS"
@  7,45 say "WIDHT"
@  7,55 say "VELOCITY"
@  8,13 say replicate("=", 4)
@  8,22 say replicate("=", 8)
@  8,35 say replicate("=", 5)
@  8,45 say replicate("=", 5)
@  8,55 say replicate("=", 8)
close all
*---------------------- Calls CHOOSING to select the crossing sites
do CHOOSING
@ 22,14 say "Press any key to continue or Press "X" to EXIT : "
@ 22,64 say "" get OPTION
read
store upper(OPTION) to OPTION
return
*==================     END  OF  SELSITE.PRG     ================*


*************************************************************************
****                      I N A R E A . P R G                        ****
*************************************************************************
* Module name : INAREA.PRG                                              *
* Purpose     : This is a sub-module of SELSITE to search the all*
*               river crossing sites in river crossing area.      *
* Called by   : SELSITE                                                 *
* Variables                                                             *
*      Local  : AREA, LOC1, LOC2                                        *
*************************************************************************

clear
*------------------------------------- Define the local variables
*----------------- AREA       : River crossing area, A or B
*----------------- LOC1, LOC2 : Coordinates of the crossing area
store space(1) to AREA
store space(8) to LOC1, LOC2
store   LEFT to LOC1
store MIDDLE to LOC2
store "A" to AREA
*----------------- TEMP contains all crossing sites in the river
*----------------- TEMP_A contains all crossing sites in area A
*----------------- TEMP_B contains all crossing sites in area B
use TEMP
index on LOC to TEMP
select 1
use TEMP_A
select 2
use TEMP_B
select 3
use TEMP index TEMP
do while .T.
    store upper(substr(LOC1,1,2)) to O
    Q = val(substr(LOC1,3,3))
    S = val(substr(LOC2,3,3))
    T = val(substr(LOC1,6,3))
    V = val(substr(LOC2,6,3))
```

```
    AX = abs ( Q - S )
    AY = abs ( T - V )
    goto top
    *------------------------- Go to the top of data file TEMP
    do while .not. eof()
        store upper(substr(LOC,1,2)) to P
        R = val(substr(LOC,3,3))
        U = val(substr(LOC,6,3))
        if O = P
            do case
                case AX < AY
                    copy to TEMP1 next 1 for ( T<=U .and. U<=V ) .or
                                             ( V<=U .and. U<=T )
                case AX = AY
                    copy to TEMP1 next 1 for ((T<=U .and. U<=V ) .or.;
                                             ( V<=U .and. U<=T)).and.;
                                             ((Q<=R .and. R<=S ) .or.;
                                             ( S<=R .and. R<=Q))
                case AX > AY
                    copy to TEMP1 next 1 for ( Q<=R .and. R<=S ) .or
                                             ( S<=R .and. R<=Q )
            endcase
            if AREA = "A"
                select TEMP_A
            else
                select TEMP_B
            endif
            append from TEMP1
        endif
        select TEMP
        skip
    enddo
    if AREA = "B"
        exit
        *---------------------------------------- Exit to RIVER.PRG
    ebdif
    AREA = "B"
    LOC1 = MIDDLE
    LOC2 = RIGHT
enddo
return
*==================     END  OF  INARE.PRG     ==================*


*********************************************************************
****                   C H O O S I N G . P R G                   ****
*********************************************************************
* Module name : CHOOSING.PRG                                        *
* Purpose     : This is a sub-module of SELSITE to select the       *
*               river crossing sites in area A and B.               *
* Called by   : SELSITE                                             *
* Variables                                                         *
*      Local  : AREA, MBRGE, MRAFT, MLINE                           *
*********************************************************************

set talk off
*------------------------------------- Define the local variables
*----------------- AREA   : River crossing area A and B
*----------------- MBRGE  : Number of bridge sites
*----------------- MRAFT  : Number of raft sites
*----------------- MLINE  : Line control variable
store 'A' to AREA
store A_BRG to MBRG
stroe A_RAFT to MRAFT
store 10 to MLINE
*----------------- SELECTED contains the selected crossing sites
use SELECTED
delete all
pack
do while .T.
```

```
*------------------------------------ Define the working area
select 1
use SELECTED
select 2
if AREA = "A"
   use TEMP_A
else
   use TEMP_B
endif
*----- TEMP_C contains the sorted crossing sites in area A or B
sort to TEMP_C on WIDTH/A
select 3
use TEMP_C
do while .not. eof()
   if MBRG <> 0
      do while .not. eof()
         *-------------------------- Select the bridge sites
         if VELO <= 2.0 .and. WIDTH >= 1.0
            @ MLINE,15 say AREA
            @ MLINE,22 say LOCA
            @ MLINE,36 say "BRGE"
            @ MLINE,44 say WIDTH
            @ MLINE,49 say "M"
            @ MLINE,55 say VELO
            @ MLINE,60 say "MPS"
            select SELECTED
            append blank
            replace LOC with TEMP_C->LOC, MEANS WITH "BRGE",;
               VELO WITH TEMP_C->VELO, WIDTH WITH TEMP_C->WIDTH
            MBRGE = MBRGE - 1
            *--------------- Bridge site can be heavy raft site
            if MRAFT <> 0
               append blank
               replace LOC with TEMP_C->LOC, MEANS WITH "BRGE",;
                  VELO WITH TEMP_C->VELO, WIDTH WITH TEMP_C->WIDTH
               MLINE = MLINE + 1
               select TEMP_C
               @ MLINE,15 say AREA
               @ MLINE,22 say LOCA
               @ MLINE,36 say "H.RAFT"
               @ MLINE,44 say WIDTH
               @ MLINE,49 say "M"
               @ MLINE,55 say VELO
               @ MLINE,60 say "MPS"
               delete
               pack
               MRAF = MRAF - 1
            endif
            exit
         endif
         select TEMP_C
         skip
      enddo
   else
      do while .not. eof() .and. MRAFT > 0
         *------------------------------ Select the raft sites
         if VELO <= 2.0 .and. WIDTH >= 1.0
            @ MLINE,15 say AREA
            @ MLINE,22 say LOCA
            @ MLINE,35 say "L.RAFT"
            @ MLINE,44 say WIDTH
            @ MLINE,49 say "M"
            @ MLINE,55 say VELO
            @ MLINE,60 say "MPS"
            select SELECTED
            append blank
            replace LOC with TEMP_C->LOC, MEANS WITH "BRGE",;
               VELO WITH TEMP_C->VELO, WIDTH WITH TEMP_C->WIDTH
            MRAFT = MRAFT - 1
            MLINE = MLINE + 1
```

```
                endif
                select TEMP_C
                skip
            enddo
        endif
        if MRAFT = 0
            use
            exit
        endif
        MLINE = MLINE + 1
    enddo
    if AREA = "B"
        exit
        *------------------------------------ Exit the main loop
    endif
    MLINE = MLINE + 1
    if A_BRG <> 0 .or. A_RAFT <> 0
        @ MLINE,13 say replicate("-",50)
    endif
    MLINE = MLINE + 2
    AREA  = "B"
    MBRG  = B_BRG
    MRAFT = B_RAFT
    close all
enddo
*------------------------------------------- End of main loop
close all
return
*==================     END  OF  CHOOSING.PRG    ================*


*************************************************************************
****                    U N I T E Q P S . P R G                      ****
*************************************************************************
* Module name : UNITEQPS.PRG                                            *
* Purpose     : This is a sub-module of POSSIBLE to examine the         *
*               organized engineer unit and their equipments.           *
* Called by   : POSSIBLE                                                *
* Variable                                                              *
*    Local    : MLINE : Line control variable                           *
*************************************************************************

set talk off
clear
use ASSIGN
*------------------ TEMP contains the all organozed engineer units
copy to TEMP fields UNIT for BELONG = MUNIT
use EQPAVAIL
*---------------------------------------- Define the working area
select A
use TEMP
select B
use EQPAVAIL index EQPAVAIL
*--------------------- TEMP1 contains the organized engineer units
*--------------------- and their equipments.
join with TEMP to TEMP1 for B-> UNIT = A-> UNIT;
      fields UNIT, AFB, LTR, M4T6
select A
use TEMP1
MLINE = 1
@  3,14 to 5,65
set color to R+
@  4,18 say "ATTACHED  ENGINEER  UNITS  AND  EQUIPMENTS"
set color to W
@  9,19 say "U N I T"
@  9,34 say "AFB"
@  9,45 say "LTR"
@  9,56 say "M4T6"
@ 10,19 say replicate("=", 7)
@ 10,34 say replicate("=", 3)
```

```
@ 10,45 say replicate("=", 3)
@ 10,56 say replicate("=", 4)
do while .not.eof()
    @ MLINE+11,19 say UNIT
    @ MLINE+11,34 say AFB
    @ MLINE+11,45 say LTR
    @ MLINE+11,56 say M4T6
    MAFB  = MAFB  + AFB
    MLTR  = MLTR  + LTR
    MM4T6 = MM4T6 + M4T6
    MLINE = MLINE + 1
    skip
enddo
@ MLINE+13,19 say "TOTAL"
@ MLINE+13,34 say str(MAFB,3,1)
@ MLINE+13,45 say str(MLTR,3,1)
@ MLINE+13,56 say str(MM4T6,3,1)
set color to R+
@ 23,15 say "Press any key to continue or Press "X" to EXIT : "
set color to W
@ 23,64 "" get OPTION
read
store upper(OPTION) to OPTION
return
*=================     END  OF  UBITEQPS.PRG     ================*


*********************************************************************
****                    C A L C E Q P S . P R G                  ****
*********************************************************************
* Module name : CALCEQPS.PRG                                       *
* Purpose     : This is a sub-module of POSSIBLE to calculate the*
*               required amount of equipments.eir equipments.     *
* Called by   : POSSIBLE                                           *
* Variable                                                         *
*     Local   : RAFTS                                              *
*********************************************************************

clear
set talk off
*-------------------- RAFTS  :  Number of rafts at each raft sites
store 0 to RAFTS
use SELECTED
do while .not. eof()
    if MEANS = "BRGE"
        RM4T6 = RM4T6 + WIDTH / 43.2
    else
        do case
            case WIDTH <= 100
                  RAFTS = 1
            case 100 < WIDTH .and. WIDTH <= 200
                  RAFTS = 2
            case 200 < WIDTH .and. WIDTH <= 300
                  RAFTS = 3
            case 300 < WIDTH
                  RAFT = 4
        endcase
        if MEANS = "H.RAFT"
            RM4T6 = RM4T6 + RAFTS / 2
        else
            RLTR  = RLTR  + RAFTS * 1
        endif
    endif
    skip
enddo
return
*=================     END  OF  CALCEQPS.PRG     ================*
```

6. Support Units Module

```
*********************************************************************
****                    S U P P O R T . P R G                    ****
*********************************************************************
* Module name : SUPPORT.PRG                                         *
* Purpose     : This is a SUPPORT module to examine available       *
*               supporting engineer units and their equipments.     *
* Called by   : RIVER                                               *
* Variables                                                         *
*      Local  : SAFB, SLTR, SM4T6, HIGHER, MLINE                    *
*               SSAFB, SSLTR, SSM4T6                                *
*********************************************************************

clear
set talk off
*-------------------------------------- Define the local variables
*----------------- SAFB,SSAFB    : Lack of AFB
*----------------- SLTR,SSLTR    : Lack of LTR
*----------------- SM4T6,SSM4T6  : Lack of M4T6
*----------------- HIGHER        : Name of higher command
*----------------- MLINE         : Line control variable
store 0 to SAFB, SLTR, SM4T6, SSAFB, SSLTR, SSM4T6
store space(7) to HIGHER
store 6 to MLINE
*-------------------------------------- Define the working area
select 1
use ASSIGN index ASSIGN
select 2
use EQPAVAIL
select ASSIGN
find &MUNIT
if found()
    HIGHER = BELONG
    *--------------------- TEMP contains the all units that assigned
    *--------------------- under higher command.
    copy to TEMP fields UNIT for BELONG = HIGHER .and. UNIT<>MUNIT
    select 3
    use TEMP
    select ASSIGN
    *--------------------- ALLENG contains the all engineer units
    *--------------------- which is assigned higher command.
    join with TEMP to ALLENG for TEMP->UNIT = ASSIGN->BELONG;
                fields UNIT
    select 4
    use ALLENG
    select EQPAVAIL
    *--------------------- ENGUNIT contains the all engineer units
    *--------------------- and equipments under higher command.
    join with ALLENG to ENGUNIT for ALLENG->UNIT = EQPAVAIL->UNIT;
          fields UNIT, AFB, LTR, M4T6
    @  1,12 to 3,65
    set color to R+
    @  2,16 say "AVAILABLE   ENGINEER   UNITS   AND   EQUIPMENTS"
    set color to W
    @  4,17 say "U N I T"
    @  4,33 say "AFB"
    @  4,44 say "LTR"
    @  4,56 say "M4T6"
    @  5,17 say replicate("=", 7)
    @  5,32 say replicate("=", 3)
    @  5,44 say replicate("=", 3)
    @  5,56 say replicate("=", 4)
    select 3
    use ENGUNIT
    do while .not. eof()
        @ MLINE,17 say UNIT
        @ MLINE,32 say AFB
```

```
        @ MLINE,44 say LTR
        @ MLINE,56 say M4T6
        MLINE = MLINE + 1
        SAFB  = SAFB  + AFB
        SLTR  = SLTR  + LTR
        SM4T6 = SM4T6 + M4T6
        skip
    enddo
    MLINE = MLINE + 1
    @ MLINE,17 say "SUBTOTAL"
    @ MLINE,32 say str(SAFB,3,1)
    @ MLINE,44 say str(SLTR,3,1)
    @ MLINE,56 say str(SM4t6,3,1)
    @ MLINE+1,17 say "HOLDING"
    @ MLINE+1,32 say str(MAFB,3,1)
    @ MLINE+1,44 say str(MLTR,3,1)
    @ MLINE+1,56 say str(MM4T6,3,1)
    TAFB  = MAFB  + SAFB
    TLTR  = MLTR  + SLTR
    TM4T6 = MM4T6 + SM4T6
    @ MLINE+2,16 say replicate("-",44)
    @ MLINE+3,17 say "TOTAL"
    @ MLINE+3,32 say str(TAFB,3,1)
    @ MLINE+3,44 say str(TLTR,3,1)
    @ MLINE+3,56 say str(TM4T6,3,1)
    @ MLINE+4,17 say "REQUIRED"
    @ MLINE+4,32 say str(RAFB,3,1)
    @ MLINE+4,44 say str(RLTR,3,1)
    @ MLINE+4,56 say str(RM4T6,3,1)
    @ MLINE+5,16 say replicate("-",44)
    SAFB  = TAFB  + RAFB
    SLTR  = TLTR  + RLTR
    SM4T6 = TM4T6 + RM4T6
    if SAFB >= 0
        SSAFB = 0.0
    else
        SSAFB = abs(SAFB)
    endif
    if SLTR >= 0
        SSLTR = 0.0
    else
        SSLTR = abs(SLTR)
    endif
    if SM4T6 >= 0
        SM4T6 = 0.0
    else
        SSM4T6 = abs(SM4T6)
    endif
    @ MLINE+7,17 say "LACK"
    @ MLINE+7,32 say str(SSAFB,3,1)
    @ MLINE+7,44 say str(SSLTR,3,1)
    @ MLINE+7,56 say str(SSM4T6,3,1)
else
    @ 12,17 say str(MUNIT) + "DOES NOT EXIST !!"
endif
set color to R+
if SAFB >= 0 .and. SLTR >= 0 .and. SM4T6 >= 0
    @ 20,18 say "Operation is possible, if you get support."
else
    @ 20,19 say "Equipments are not enough for operation."
endif
@ 23,15 say "Press any key to continue or Press "X" to EXIT : "
set color to W
@ 23,64 say "" get OPTION
read
store upper(OPTION) to OPTION
return
*================     END  OF  SUPPORT.PRG     ================*
```

7. Generate Module

```
*********************************************************************
****                  G E N E R A T E . P R G                    ****
*********************************************************************
* Module name : GENERATE.PRG                                        *
* Purpose     : This is a GENERATE module to estimate the fini-     *
*               shing time of river crossing operation.             *
* Called by   : RIVER                                               *
* Calls       : TRIPRAFT, CALCATM                                   *
* Variables                                                         *
*      Local  : HV_HRAFT, HV_BRGE, LV_LTR, LV_HRAFT, LV_BRGE,       *
*               HV_TOTAL, LV_TOTAL, AT_HRAFT, AT_BRGE, AT_LTR,      *
*               HV_HOUR, LV_HOUR, HOUR, NOHRAFT, NOBRGE, NOLTR,     *
*               TRIPS, H_VEH, L_VEH                                 *
*********************************************************************

clear all
set talk off
*-------------------------------------- Define the local variables
*--------- HV_HRAFT : Number of heavy vehicles crossed by HV.raft
*--------- HV_BRGE  : Number of heavy vehicles crossed by bridge
*--------- LV_LTR   : Number of light vehicles crossed by LTRdge
*--------- LV_HRAFT : Number of light vehicles crossed by HV.raft
*--------- LV_BRGE  : Number of light vehicles crossed by bridge
*--------- HV_TOTAL : Total number of crossed heavy vehicles
*--------- LV_TOTAL : Total number of crossed light vehicles
*--------- AT_HRAFT : Heavy raft assembly time
*--------- AT_BRGE  : Bridge assembly time
*--------- AT_LTR   : LTR assembly time
*--------- HV_HOUR  : Heavy vehicle crossing time
*--------- LV_HOUR  : Light vehicle crossing time
*--------- HOUR     : Heavy or light vehicle crossing time
*--------- NOHRAFT  : Number of heavy rafts
*--------- NOBRGE   : Number of bridges
*--------- NOLTR    : Number of LTR
*--------- TRIPS    : Number of trips per hour
*--------- H_VEH    : Total number of heavy vehicles
*--------- L_VEH    : Total number of light vehicles
store 0 to HV_HRAFT, HV_BRGE, LV_LTR, LV_HRAFT, LV_BRGE,;
           HV_TOTAL, LV_TOTAL
store 00.0 to AT_HRAFT, AT_BRGE, AT_LTR, HV_HOUR, LV_HOUR
store 0 to NOHRAFT, NOBRGE, NOLTR, TRIPS, H_VEH, L_VEH
*------------------- Calls TRIPRAFT to calculate the trips per hour
*------------------- and number of rafts.
do TRIPRAFT with TRIPS, NOFRAFT, NOLTR
*------------------- Calls CALCATM to calculate the assembly time
do CALCATM with AT_HRAFT, AT_BRGE, AT_LTR, NOBRGE
do while HV_HOUR <= AT_HRAFT .or. HV_HOUR < AT_BRGE
    if HV_HOUR > AT_HRAFT .and. NOHRAFT <> 0
       HV_HRAFT = HV_HRAFT + ( TRIPS * NOHRAFT )
    endif
    HV_TOTAL = HV_TOTAL + HV_HRAFT
    HV_HOUR  = HV_HOUR  + 1
enddo
use DIVEQPS index DIVEQPS
seek MUNIT
H_VEH = H_EQPS
L_VEH = L_EQPS
do while HV_TOTAL < H_VEH
    *--------------------------- Calculate crossed heavy vehicles
    HV_HRAFT = HV_HRAFT + ( TRIPS * NOHRAFT )
    HV_BRGE  = HV_BRGE  + ( 200 * NOBRGE )
    HV_TOTAL = HV_TOTAL + HV_HRAFT + HV_BRGE
    HV_HOUR  = HV_HOUR  + 1
enddo
do while AT_LTR >= LV_HOUR
```

```
    *---------------------- Calculate crossed light vehicles by LTR
    LV_TOTAL = LV_TOTAL + LV_LTR
    LV_HOUR  = LV_HOUR + 1
enddo
do while LV_TOTAL < L_VEH
    *---- Calculate crossed light vehicles by LTR, BRGE and HV.raft
    LV_LTR = LV_LTR + ( TRIPS * NOLTR )
    if HV_HOUR <= LV_HOUR
        LV_HRAFT = LV_HRAFT + ( TRIPS * NOHRAFT * 2 )
        LV_BRGE  = LV_BRGE  + ( 200 * NOBRGE )
    endif
    LV_TOTAL = LV_TOTAL + LV_LTR + LV_HRAFT + LV_BRGE
    LV_HOUR  = LV_HOUR  + 1
enddo
if HV_HOUR > LV_HOUR
    HOUR = HV_HOUR
elsr
    HOUR = LV_HOUR
endif
@  6,20 to 8,54
set color to W
@  7,24 say "EXPECTED   FINISHING   TIME"
set color to W
@ 11,24,say "HEAVY    VEHICLE  :  H + "
@ 11,48,say ltrim(str(HV_HOUR ))
@ 13,24,say "LIGHT    VEHICLE  :  H + "
@ 13,48,say ltrim(str(LV_HOUR ))
set color to R+
@ 16,24 say "FINISHING   TIME  :  H + "
@ 16,48 say ltrim(str(HOUR))
@21,13 say "End of running, Press any key to back input mode : "
set color to W
@ 21,63 "" get OPTION
read
store upper(OPTION) to OPTION
return
*=================     END  OF GENERATE.PRG     =================*



*************************************************************************
****                    T R I P R A F T . P R G                      ****
*************************************************************************
* Module name : TRIPRAFT.PRG                                            *
* Purpose     : This is a sub-module of GENERATE to calculate the*
*               available trips of rafts and number of rafts.    *
* Called by   : GENERATE                                                *
* Variables                                                             *
*      Local  : RAFTS, TRIP                                             *
*************************************************************************

parameters TRIPS, NOHRAFT, NOLTR
*------------------------------------ Define the local variables
*--------------------- RAFTS  : Number of rafts
*--------------------- TRIP   : Available number of trips per hour
store 0 to RAFTS, TRIP
use SELECTED
do while .not. eof()
    do case
        case WIDTH <= 100
             RAFTS = 1
             TRIP  = 8
        case 100 < WIDTH .and. WIDTH >= 200
             RAFTS = 2
             TRIP  = 6
        case 200 < WIDTH .and. WIDTH >= 300
             RAFTS = 3
             TRIP  = 4
        case 300 < WIDTH .and. WIDTH >= 500
             RAFTS = 4
```

```
            TRIP  = 2
    endcase
    if MEANS   = "H_RAFT"
       NOHRAFT = NOHRAFT + RAFTS
    endif
    if MEANS   = "L_RAFT"
       NOLTR   = NOLTR + RAFTS
    endif
    if MEANS   = "L_RAFT" .or. H_RAFT
       TRIPS   = TRIPS + TRIP
    endif
    TRIPS = TRIP / ( A_RAFT + B_RAFT )
    skip
enddo
close all
return
*=================== END  OF  TRIPRAFT.PRG     ==================*



*********************************************************************
****                    C A L C A T M . P R G                    ****
*********************************************************************
* Module name : CALCATM.PRG                                         *
* Purpose     : This is a sub-module of GENERATE to calculate the*
*               assembly time og bridge and rafts.                  *
* Called by   : GENERATE                                            *
* Variables                                                         *
*      Local  : SETS, SSET, TIME                                    *
*********************************************************************
parameters A_HRAFT, AT_BRGE, AT_LTR, NOBRGE
clear
set talk off
*------------------------------------ Define the local variables
*----------------- SETS   : Sets of equipments
*----------------- SSETS  : Sets of equipments
*----------------- TIME   : Finishing time
store 0 to SETS, SSET, TIME
*------------------------------------- Define the working area
select 1
use CROSSEQP
do while .not. eof()
    if MEANS  = "LTR"
       AT_LTR = ATIME
    endif
    if MEANS = "M4T6"
       AT_HRAFT = ATIME / NORAFT
       TIME = ATIME
       select SELECTED
       do while .not. eof()
          if MEANS = 'BRGE'
             SSET  = WIDTH / 43.2
             SETS  = SETS + SSET
             NOBRGE = NOBRGE + 1
          endif
          skip
       enddo
       AT_BRGE = TIME * SETS/ NOBRGE
    endif
    select CROSSEQP
    skip
enddo
return
*====================     END  OF  CALCATM     ==================*
```

LIST OF REFERENCES

1. Headquarters, Department of the Army, *Operation*(*FM 100-5*), Washington, DC., 1982

2. Headquarters, Department of the Army, *Staff Officers Field Manual*(*FM 101-5*), Washington, DC., 1972

3. Headquarters, Department of the Army, *Engineer Combat Operation*(*FM 5-100*), Washington, DC., 1979

4. Headquarters, Department of the Army, *River Crossing Operation*(*FM 90-13*), Washington, DC., 1978

5. Headquarters, Department of the Army, *Obstacles*(*FM 90-7*), Washington, D.C., 1979

6. Fairley, R. E., *Software Engineering Concepts*, McGraw-Hill, Inc., 1985

7. Davis, William S., *System Analysis and Design*, Addison Wesley, Inc., 1983

8. Davis, Gordon Bitter, *Management Information Systems*, McGraw-Hill, Inc., 1985

9. Waterman, Donald Arthur, *A Guide to Expert Systems*, Addison-Wesley, Inc., 1986

10. Levin, Robert I., *A Comprehensive Guide to AI and Expert Systems*, McGraw-Hill, Inc., 1986

11. Townsend, Carl, *Mastering dBASE III Plus*, SYBEX Inc., Berkeley, CA., 1986

12. Yourdon, Edward, *Managing the Structured Techniques*, Yourdon Press Inc., New York, N.Y., 1986

13. Kroenke, David, *Database Processing*, Science Research Associates, Inc., 1983

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chief of Naval Opertions Director, Information Systems(OP-945) Navy Department Washington, D.C., 20350-2000	1
4.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5000	1
5.	Associate Proffessor C. Thomas Wu, Code 52Wq Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5000	1
6.	Library, Korea Military Academy Gong Neung-Dong, Do Bong-ku, Seoul Republic of Korea	2
7.	Library, Korea Army Engineer School Kimhae-Si, Bukboo-Dong, Kyung Nam Republic of Korea	2
8.	Department of Combat Development Korea Army Engineer School Kimhae-Si, Bukboo-Dong, Kyung Nam Republic of Korea	1
9.	Department of Planning, P.O.Box 10 Yongsan-Ku, Yongsan-Dong, Seoul Republic of Korea	2
10.	LTC Jung, Yun Su SMC #2643 Naval Postgraduate School Monterey, CA 93943	1

11.	LTC Hur, Soon Hae SMC #1654 Naval Postgraduate School Monterey, CA 93943	1
12.	Maj. Lee, Yong Moon SMC #1110 Naval Postgraduate School Monterey, CA 93943	1
13.	Maj. Park, Tae Yong SMC #2297 Naval Postgraduate School Monterey, CA 93943	1
14.	Cpt. Yee, Seung Hee SMC #2640 Naval Postgraduate School Monterey, CA 93943	1
15.	Cpt. Kim, Tae Woo SMC #2555 Naval Postgraduate School Monterey, CA 93943	1
16.	Jang, Chang Kun The National Assembly Library Seoul, Republic of Korea	1
17.	Jang, Chang Ki Kang Seo-Ku, Mock 4-Dong 780-10 Seoul, Republic of Korea	7

Thesis
J282 Jang
c.1 Intelligence decision support system for the Republic of Korea Army engineer operation.